Forced Entry

Description: So what happened with last week's Patch Tuesday? Was there anything of note? If we took a quick overview of just a tiny bit of last week's news, what would that look like, and what would those stories all have in common? What new developer-centric service is Google making freely available for the good of the open source community? What moves are WhatsApp making to improve the security for the world's most popular secure messaging system? What happens when a European psychotherapy clinic apparently doesn't care enough to provide even minimal security for the patients' records? And finally, in this week's deep dive, we're going to answer the question: What could researchers have found inside a piece of the NSO Group's Pegasus smartphone spyware that actually terrified them, and why?

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-919.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-919-lq.mp3

SHOW TEASE: It's time for Security Now! with Steve Gibson. I'm Jason Howell, filling in for Leo Laporte one last time, I promise you. He's going to be back next week. But we have a lot of fun talking about the security news this week. Google's free Assured Open Source Software service, Steve talks all about that. Microsoft Patch Tuesday kind of feels like Patch Tuesday in perpetuity. And in reality that's probably exactly what it is. We check in on a European psychotherapy clinic that failed to provide minimal security for their patients' records, and the consequences attached to doing so. And Steve Gibson takes a closer look at the ForcedEntry exploit for iPhones. It's really fascinating. That's next on Security Now!.

JASON HOWELL: This is Security Now! with Steve Gibson, Episode 919, recorded Tuesday, April 18th, 2023: Forced Entry.

It's time for Security Now! with Steve Gibson and - Leo is still not here. I'm Jason Howell, sitting in for Leo. But Steve, you are always the constant on this show, and we appreciate that about you.

Steve Gibson: Glad to be back with you, Jason. Glad you're here.

JASON: Thank you.

Steve: Holding down the fort at your end. And yes.

JASON: Doing what I can.

Steve: Certainly glad to be back. Okay. So an interesting episode, I think, for us. This is Security Now! #919 for the 18th, of April; right? I got the month right this time, or whatever it was I got wrong last time. Anyway, okay. So we're going to, as we have

been, answer some questions. What happened with last week's Patch Tuesday? Was there anything of note? If we took a quick overview of just a tiny bit of last week's news, just sort of a quick scan, what would that look like, and what would all those stories have in common? What new developer-centric service is Google making freely available for the good of the open source community? And I'm very bullish about this. It looks really cool.

What moves are WhatsApp making to improve the security of what is the world's most popular secure messaging system, you know, theirs? What happens when a European psychotherapy clinic apparently doesn't care enough to provide even minimal security for their patients' records? And finally, in this week's deep dive, we're going to answer the question: What could researchers have found - these are Google Project Zero researchers who, you know, they've been around the block a few times; right? You know, they've seen it all. What could they have found inside a piece of the NSO Group's Pegasus smartphone spyware that actually and truly terrified them? And why?

JASON: That's intriguing. They've seen stuff.

Steve: And what they found, they're like, oh, goodness.

JASON: What they found may surprise you, as clickbait headlines might have. You're just going to have to wait to find out. That's what we know.

Picture of the Week time. This one's not too difficult to decipher. You look at it for a second, you figure it out. You wonder, did that really happen? Really? Somebody did that? At least that was my experience.

Steve: Okay. So for those who don't have the advantage of video or the show notes, we've actually had a variation on this before. There was one where bolt cutters were being held captive at a hardware store with like a thick gauge steel cable that was looped through them. But again, it's a bolt cutter that you're trying to keep from being stolen, obviously, by looping something that this thing was designed to cut through its jaws. And so we have a variation on that this week. I thank one of my Twitter followers for sending it. Here we have some needle-nose pliers where down near the hinge of the jaw they've got wire cutters. And, I mean, this doesn't even take any imagination. You don't have to move anything. You just have to squeeze the handle on one of these because the thin metal cord, essentially, that is looped through them is going through the wire cutter portion of these needle-nose pliers.

JASON: I mean, it couldn't be lined up more perfectly than literally just pull the handle. All you need to do is pull the handle.

Steve: And looking at it, you know, looking at it, even if you were not a thief, and you encountered this, you'd be just tempted to squeeze the handle, just to say, okay, idiots. I'm not taking this, but you should take a lesson. Anyway, wow.

JASON: Yeah, I guess instead of looping it - well, no, I guess there's no real way to do that. If it's got a wire cutter there, you've just got to come up with a different system.

Steve: You're in trouble. I know that you produce the podcast every week, whether you're on-camera or not. Did you see the picture of the bicycle that was locked to a yellow post that, like, had no top?

JASON: Had no top; right.

Steve: So you just lift the bicycle off.

JASON: I mean, I guess security in that case is like, can we slow down the majority of people who might do something? But that's, like, pretty low effort right there.

Steve: Or maybe an IQ test? Like, okay, you know, if you want to steal the bike, but you can't figure out how, you probably should not be riding a bicycle.

JASON: But if you can, hey, free bikes.

Steve: That's right, yeah. Wow. Okay. So last Tuesday was our monthly celebration of all things we know of - hopefully all the things we know of - that we wish were different in the details of our Windows software. I would like to say that, to that end, we received new and improved software. But since we never appear to be getting ahead, we're basically just treading water, hoping to keep our heads above, what that suggests is that we actually received last week software that is broken in new and different ways than what we had before. You know, what's different is that we haven't yet discovered all of the new ways that this replacement software is broken, but we've eliminated a bunch of the ways that we knew the old software was broken.

So we'll be back next month, and the month after that, and the month after that, ad infinitum. This will never end. Think about that. This shows no sign of ever ending. It's not as if the number of problems is diminishing over time, patch counts are dropping, and there's a chance of us not needing one of those monthly rituals. No. I'm not really up on pop culture. Leo typically helps me with these things. But I think that the official technical term for this monthly dance that ultimately goes nowhere is Whac-A-Mole.

So, yes. This month we reportedly cured 97 flaws in our software from Microsoft, including one actively exploited zero-day flaw. There were seven flaws which received a critical rating because they allowed for remote code execution. However, overall, nearly half of this month's 97 vulnerabilities allowed for remote code execution. Okay. So here's the breakdown. Of the 97, 45 were remote code execution vulnerabilities; 20 were elevation of privilege vulnerabilities; 10 information disclosure; nine denial of service; eight security feature bypass; and six were spoofing vulnerabilities.

Okay, now, as we know, the two most potent vulnerability classes are remote code execution and elevation of privilege. So together that was 45 plus 20. They formed 65 out of the total 97 patched problems. Oh, and that count is separate from the 17 newly discovered vulnerabilities which Microsoft fixed for us the week before in their Edge browser. The zero-day vulnerability is an elevation of privilege problem in the Windows Common Log File System Driver. This tends to be a problematic component of Windows. We've run across problems in it before, and we hear about it too often.

The problem with Windows EoP (elevation of privilege) bugs in core component drivers, such as the common log file driver, is that they're always there. That is, the drivers are always there, and they're in the kernel. They're installed on every Windows computer. Consequently, they allow attackers with few or no significant access privileges to promote themselves directly to the system account level, you know, root, which gives them free rein over the machine. This is why elevation of privilege, which sounds less scary than remote code execution, can be at least as troublesome, and certainly at least as handy to attackers.

One of the critically rated remote code execution bugs may be a bit extra worrisome. Microsoft disclosed a remote flaw in their Microsoft Message Queue system, known as MSMQ. It provides a robust means for intersystem message-passing communications when absolute reliability in messaging is critical. You know, you just don't send it and hope that it got there. It's got, you know, delivery confirmation and receipts and all that stuff added to it. But since it's a bit special purpose, it's not always enabled. But when it

is, it can be trouble because Microsoft gave it a 9.8 CVSS score. Which we know it's very difficult to get, I mean, rare do you get a 10; 9.8's still way bad.

And apparently from Microsoft's description they said: "To exploit this vulnerability, an attacker would need to send a specially crafted malicious MSMQ packet to a MSMQ server. This could result in remote code execution on the server side." Okay. So we've been around long enough to know that's not good. There's some exploitable buffer overflow-ish sort of thing where just sending a packet at the server can allow it to execute your code. So not good. So let's hope that MSMQ servers are entirely constrained, as they should be, within enterprise Intranets, and that none are exposed on the public Internet. On the other hand, we know reality, too.

We now know with absolute certainty that it's not only security researchers who compare pre-patched software to post-patched software in order to reverse engineer the flaws that were fixed by the patch and thus discover the vulnerability that wasn't known until the patch was made available. So it won't be long before bad guys are working to gain entry into enterprises that, A, have not yet applied the April patches to their border servers; and, B, have the MSMQ service publicly exposed where you almost sort of feel like they deserve what they're going to get because this really should not be, you know, there's no point, there's no reason for it to be out on the Internet.

And as a matter of fact, another worrisome critical bug also should not be out on the Internet. It's a remote code execution flaw in Microsoft's DHCP service. As we know, DHCP is the Dynamic Host Configuration Protocol which is the way most of the computers in the world receive their IP addresses. It's a slick system that allows a machine that knows absolutely nothing about the network it's connected to, to emit a totally generic Ethernet broadcast packet out onto whatever network it's attached to, wired or wireless. That packet, which essentially says, "Hey, I'm here, help," will be received by a listening DHCP server, which is then able to reply with all of the instructions and settings necessary to get that machine's IP stack configured with a unique, non-conflicting local IP address, the IP of the network's gateway so it's able to talk to the rest of the world, and the addresses of some DNS servers which are available to the hosts on the Intranet.

Okay. So my own server network at GRC uses 100% statically configured IP addresses because there's just no need, nothing is coming and going, you know, people are not attaching to that network on an ad hoc basis, unlike at Starbucks and airports and in our homes. At home I have a combination of static IPs and DHCP. Some machines I want to always have at the same location; whereas things like iPhones, iPads, Rokus, you know, IoT thermostats and whatnot, they don't matter, and there's no point in tying them down.

One of the cool things that a DHCP server can be configured for is the range of IPs that it is allowed to allocate from. So, for example, in a /24 network like most of us have, most residential users have, which typically begins with 192.168.0 or .1, the final byte can range from 0 to 254. So a residential network's DHCP server could be configured to automatically assign IPs only from the upper range, like where the last byte runs from 100 to 254 - or I'm sorry, 253, 254 is typically the gateway - which leaves the lower 100 IPs, 0 to 99, guaranteed to always be available for manual static assignment.

In any event, DHCP server flaws have historically and understandably been a little extra frightening because, if you stop to think about it, the entire nature of the way DHCP servers operate is inherently unauthenticated. By design, they're supposed to be there for anyone who asks for their help, without question. This means that DHCP servers must be robust enough to accept and reply to packets from unknown, untrusted, and potentially hostile devices. You know, they've got to be sturdy. Fortunately, although we've seen very serious problems in the past with DHCP servers, today's problem only rates an 8.8 out of 10 since the flaw turns out to be a post-authentication problem, so

different than just querying for your IP address and stuff. That's still bad, but at least it's less so.

Microsoft said: "An authenticated attacker could leverage a specially crafted RPC call (remote procedure call)" - so I guess that would be remote procedure call call, RPC call - "to the DHCP service to exploit this vulnerability," they said. And being DHCP, it's going to be on the LAN, not the WAN, so exploitation of this vulnerability requires an attacker first have access to the local internal network, then have some form of authentication which is recognized by the DHCP server, and then be clever enough to craft some RPC call in order to take it over.

So we did not have, this month, any serious post-patch meltdowns, which we've covered in prior months - actually, I don't think any this year, but lots were last year. Microsoft had a tough time, especially at the beginning of last year. But none of those have been reported this month. So there's no obvious reason not to proceed, update our machines to eliminate the latest batch of known problems while apparently, given all the evidence, introduce at least as many new unknown problems. Which, you know, we'll get to fixing eventually. So we'll see you next month for more of the same.

Okay. So I was perusing the news of the past week, right, to pick out those things that I thought were most interesting and worthy of some conversation. There's a section which Risky Business News calls "Breaches and Hacks." And as I just scanned it, none of them were gripping, none of them I wanted to go into in any detail. But still, they just sort of had me shaking my head. I remind myself that our listeners are only aware, at least through this podcast, of those things that rise to the top of the week's news which, you know, seem worthy enough for mention, you know, and some further exploration and, you know, deeper dives here. But that can have the effect of inadvertently obscuring what the broader view looks like of just how much nonsense is continually going on out there in the wider world.

So Risky Business's tag line is "It's a Jungle Out There." So I wanted to share briefly what I encountered in just this one short section covering the past week's news under the topic of "Breaches and Hacks." So I'm just going to read the headline and the short little blurb, just to give you a sense for it.

So NCR gets ransomwared: "NCR, the world's largest banking and payments software maker, has confirmed that a recent data center outage was caused by a ransomware attack. According to NCR, the incident has impacted the availability of its Aloha POS payments platform. The intrusion was claimed by the AlphV ransomware gang, which temporarily listed the company on its dark website."

Also, cyberattack on irrigation systems: "A cyberattack is suspected of having disrupted the operations of irrigation systems in Israel's Upper Galilee region. Water controllers for irrigating fields in the Jordan Valley and control systems for the Galil Sewage Corporation were down last week during yearly cyberattacks aimed at Israeli targets known as OpIsrael. While several hacktivist groups announced their participation in OpIsrael this year, the identity of the attacker remains unknown, the Jerusalem Post reported."

Also, Hyundai data breach. "Hyundai's Italian and French branches have leaked customers' details. Exposed data includes email addresses, telephone numbers, physical addresses, and vehicle chassis numbers." MSI ransomware attack: "Taiwanese hardware vendor MSI has confirmed a security breach after the Money Message ransomware gang claimed to have breached and encrypted some of the company's systems."

Hundred Finance crypto heist: "The Hundred Finance platform was hacked for $7.4 million following a flash-loan attack." GDAC crypto heist: "South Korean cryptocurrency platform GDAC was hacked earlier this month for $13 million worth of assets."

Bitrue crypto heist: "The Bitrue cryptocurrency exchange says it was hacked and lost $23 million worth of assets following an attack on one of its hot wallets on April 14th. Bitrue says it's currently investigating the incident and doesn't know how the attack took place."

Yearn Finance crypto heist: "A threat actor exploited a bug in the Yearn Finance platform to steal $11.6 million worth of cryptocurrency assets." And finally, Terraport crypto heist: "The Terraport DeFi platform was hacked for $4 million worth of crypto assets."

Okay. What a week. 19 years ago, when Leo first proposed this podcast to me, none of this sort of nonsense was going on. You know, we had cute little email macro viruses that liked to send out email to all of the contacts in someone's address book. You'd receive a bizarre email from your mom and call her up and say "Uh, Mom. Unless we have a Nigerian cousin that you've never told me about, I think that maybe your computer has a virus." You know, and that was pretty much the extent of it; right? Simpler times.

But thinking about that brief news blurb summary that I just shared, you know the one thing that most of them have in common? The thread that winds and weaves through them all is cryptocurrency. That's today's common factor. Partly it's because it enables ransoms, payments which are now facilitated by the inherent anonymity offered by cryptocurrency. But even more so, it's the cryptocurrency itself. The software being used to implement cryptocurrency systems is apparently, given the evidence, no better than the software that does anything else. But rather than just getting the font wrong when an item is selected in a word processor, when a hole is found in cryptocurrency software, millions of dollars of actual tradable fiat currency drain out through that hole. And where there's money, there's bad guys.

JASON: That's exactly it. Where there's money, there's bad guys. Yup.

Steve: Yeah. Okay. So this is cool. We're in need of some good news at this point, and we have some courtesy of Google. Last Wednesday the 12th, Google announced that their so-called Google Assured Open Source Software service would now be generally available to all. Here's what Google posted and explained.

They said: "Threats to the software supply chain and open source software continue to be major areas of concern for organizations creating apps and their developers. According to Mandiant's M-Trends 2022 report" - so from last year - "17% of all security breaches start with a supply chain attack, the initial infection vector second only to exploits."

They said: "Building on Google's efforts to improve open source security, we're announcing the general availability of the Assured Open Source Software" - which they call Assured OSS - "service for Java and Python ecosystems. Available today at no cost, Assured OSS gives any organization that uses open source software the opportunity to leverage the security and experience Google applies to open source dependencies by incorporating the same open source software packages that Google secures and uses in their own developer workflows."

Wow. Okay. So in other words, just to be clear, this new service will provide free use of the same vetted and carefully reviewed Java and Python packages that Google themselves use internally for their own applications. Google explained: "Using Assured OSS, organizations can obtain their OSS packages from a trusted and known supplier; know more about their ingredients with Assured Software Bill of Materials provided in industry standard formats like SPDX and VEX; reduce risk as Google is actively scanning, finding, and fixing new vulnerabilities in curated packages; increase confidence in the integrity of the ingredients they're using through signed, tamper-evident provenance; and choose from more than 1,000 of the most popular Java and Python packages, including common machine learning and AI projects like TensorFlow and Pandas.

"Since our public preview announcement in May of last year (2022), and integrating Assured OSS as a key component in Software Delivery Shield the following October, we've received an overwhelmingly positive response and interest from our customers." They then cite one. Jon Meadows, managing director and Citi Tech Fellow, Cyber Security at Citi said: "Citi has been an advocate and active leader in the industry's efforts to secure enterprise software supply chains. Both Citi and Google see untrusted and unverified open source dependencies as a key risk vector. This is why we've been excited to be an early adopter of Google Cloud's new Assured OSS product. Assured OSS can help reduce risk and protect open source software components commonly used by enterprises like us."

Okay. So Google said: "Assured OSS guards OSS packages against attacks and risk by continuously mirroring key external ecosystems to manage end-to-end security without creating forks; managing the security and integrity of the mirrored repos and end-to-end build tool chain with tamper-evident provenance and attestations; continuously scanning for, fuzz testing, and fixing critical vulnerabilities, which are then quickly contributed back upstream to limit the exposure time and blast radius; and, finally, operating a critical patching team to support covered packages."

Another early adopter of this, of Google's, is Melinda Marks, a senior analyst with ESG. She was quoted saying: "As organizations increasingly utilize OSS for faster development cycles, they need trusted sources of secure open source packages. Without proper vetting and verification or metadata to help track open source software access and usage, organizations risk exposure to potential security vulnerabilities and other risks in their software supply chain. By partnering with a trusted supplier, organizations can mitigate these risks and ensure the integrity of their software supply chain to better protect their business applications."

Okay. And then Google finishes: "There are significant security benefits to Assured OSS adopters and the larger community from the curation process. Since our Assured OSS team curated the first 278 packages, we have been the first to find 48% of the new vulnerabilities (CVEs). Each of these CVEs has been fixed and upstreamed."

So to me, this seems like all good news. I do have the sense for, like, those who have a cross to bear, an axe to grind with Google, that this might cause free software open source purists to blow a gasket if they sense any sort of effort to in any way privatize or taint the openness and freeness of open source software. But we all know too well now that public open source software registries are under constant and growing attack. They are being polluted with similarly named, deliberately malicious packages, and people are downloading them. So while the idea of having an open community of like-minded contributors is terrific, the reality is that not everyone is like-minded. So I, for one, think that it makes all kinds of sense to have access to a vetted and curated source of open source software. So to Google I say "Bravo." I think this is a big win.

JASON: All right. WhatsApp? What's happening with WhatsApp? What's happening?

Steve: I'm not a big social media user. As Leo often notes, it took him quite a while to get me into Twitter. And he's not really even trying to move me over to Discord. I think he knows that's not going to happen. I just, you know, I don't need - I can't handle one more outlet that I need to be checking all the time.

JASON: I understand, Steve. Totally get it.

Steve: Okay. And also the fact is I'm not moving state secrets securely. And of course I'm skeptical that it's actually possible to do so with any of our modern mainstream technologies. We've often joked that if you really want to tell someone a secret, go out into the middle of a football field, both of you huddle under a thick blanket, and whisper

into each other's ear. Oh, and you have to be naked so that there's no recording devices anywhere on either of you. And then you have a chance of actually conveying a secret and not having it leak out anywhere. Otherwise, eh, I don't think so. Anyway, the only encrypted messaging service I bother with is iMessage, and it's just because it's there. It really wouldn't matter much to me if it wasn't encrypted. I have some friends who have green bubbles on my iPhone, which means they're using SMS. And, you know, those messages fly out in the clear.

Okay. But all that aside, when the world's leading secure messaging platform - which for better or for worse is WhatsApp - announces security improvements, it's something that we need to at least take note of. So last week WhatsApp received three new security features. They called them Account Protect, Device Verification, and Automatic Security Codes. And we'll look at these things one at a time. So here's how WhatsApp described the first one of those, Account Protect.

They said: "If you need to switch your WhatsApp account to a new device, we want to double-check that it's really you." Okay, good. They said: "From now on, we may ask you on your old device to verify that you want to take this step as an extra security check." To which I thought, you're not doing that already? That seems like a really good thing to do. Anyway, they said: "This feature can help alert you to an unauthorized attempt to move your account to another device." As I said, that one seems kind of obvious. We often see that when we're changing an email account, a notification is sent to both the old and the new accounts to help prevent a malicious email change. This is the same sort of thing. And sort of similarly, when a new trust relationship is being established, both ends are typically asked to affirm that they both wish to trust the other.

So this so-called Account Protect feature is obviously useful, but you have to wonder why it's only happening now. And we know, right, that anything that is done to increase security will have some degree of backlash, some trouble. You know, people have moved their WhatsApp account from device to device in the past, and the old device never asked if they wanted to lose access to it in favor of another one. Now it's going to. So, okay. That'll cause some confusion. But it sure does seem like it's worth the additional security of not, you know, suddenly discovering that WhatsApp has moved to a different device that you did not authorize.

Okay, next up we have what WhatsApp is calling Device Verification. Describing this, they write: "Mobile device malware is one of the biggest threats to people's privacy and security today because it can take advantage of your phone without your permission and use your WhatsApp to send unwanted messages. To help prevent this, we have added checks to help authenticate your account, with no action needed from you, and better protect you if your device is compromised. This lets you continue using WhatsApp uninterrupted."

Okay. So I had to do a little more digging into that one. This Device Verification is some backend technology designed to prevent malware which crawls into a user's phone from obtaining the user's authentication token and then impersonating them in subsequent secure messages. Okay, now having said that, that's not actually what it does. But exactly what's going on here isn't yet clear.

So here's how WhatsApp describes the situation. They said: "WhatsApp uses several cryptographic keys to ensure that communications across the app are end-to-end encrypted. One of these is the authentication key, which allows a WhatsApp client to connect to the WhatsApp server to reestablish a trusted connection. This authentication key allows people to use WhatsApp without having to enter a password, PIN, SMS code, or other credential every time they turn on the app. This mechanism is secure because the authentication key cannot be intercepted by any third party, including WhatsApp. If a device is infected with malware, however, the authentication key can be stolen." Okay,

so the point is that's different than using it locally on the app and, like, sending out messages through the client's WhatsApp app. This is obtaining the authentication key and exfiltrating it.

So they said: "We are primarily concerned about the popularity of unofficial WhatsApp clients that contain malware designed for this purpose. These unofficial apps put users' security at risk, and it is why we encourage everyone using WhatsApp to use the official WhatsApp app. Once malware is present on user devices, attackers can use the malware to capture the authentication key and then use it to impersonate the victim to send spam, scams, phishing attempts, et cetera, to other potential victims. Device Verification," as they're calling this, "will help WhatsApp identify these scenarios and protect the user's account without interruption."

Okay, now, I'll just note, before we go any further, that WhatsApp, in writing this, is being optimistic, if not disingenuous, since serious spy malware, such as we will be looking at closely in our final topic today, is purpose-designed to do this with the official WhatsApp app. So they say, oh, yeah, it's only the third-party apps that are like, you know, third-party WhatsApp apps that are a problem. Clearly, the vast majority of WhatsApp users are using the WhatsApp app, and that's what the spyware's targeting. Okay. In any event, the problem is certainly real. So here's how they're solving it.

They wrote: "WhatsApp has built Device Verification to benefit from how people typically read and react to messages sent to their device. When someone receives a message, their WhatsApp client wakes up and retrieves the offline message from WhatsApp's server. This process cannot be impersonated by malware that steals the authentication key and attempts to send messages from outside the user's device." Okay. That's not quite true, but when we understand what they're doing, we'll be able to sort of dissect this.

They wrote: "Device Verification introduces three new parameters: a security-token that's stored on the user's device; a nonce that is used to identify if a client is connecting to retrieve a message from WhatsApp server; and an authentication challenge that is used to asynchronously ping the user's device." They said: "These three parameters help prevent malware from stealing the authentication key" - again, that's not what it does, but okay - "and connecting to WhatsApp server from outside the user's device. Every time someone retrieves an offline message, the security token is updated to allow seamless reconnection attempts in the future. This process is called 'bootstrapping the security token.'" Okay. And giving it a fancy name doesn't make it fancy.

"Every time a WhatsApp client connects to the WhatsApp server, we require the client to send us the security token that's on their device. This allows us to detect suspicious connections from malware that's trying to connect to the WhatsApp server from outside the user's device. An authentication challenge is an invisible ping from the WhatsApp server to a user's device. We only send these challenges on suspicious connections.

"There are three possible responses to the challenge. Success: The client responds to the challenge from the connecting device. Failure: The client responds to the challenge from a different device. This means the connection being challenged is very likely from an attacker, and the connection will be blocked; or, finally, no response: The client doesn't respond to the challenge. This situation is rare and indicates that the connection being challenged is suspicious. We retry sending the challenge a few more times. If the client still doesn't respond, the connection will be blocked."

Okay. And then they finish by adding: "Device Verification has been rolled out to 100% of WhatsApp users on Android and is in the process of being rolled out to iOS users. It enables us to increase our users' security without interrupting their service or adding an additional step they need to take. Device Verification will serve as an important and

additional tool at WhatsApp's disposal to address rare key theft security challenges. We will continue to evaluate new security features to protect the privacy of our users."

Okay. So here's what's actually going on. What they've described, when you sort through and read out the mumbo-jumbo, is a simple, but certainly useful protocol for detecting when two physically separate clients are both checking in for messages. The idea is that every WhatsApp client will now be maintaining some state, a transient state, in the form of this new token which they have received from the WhatsApp server. It's probably just a random nonce. That's all it needs to be. It was the random nonce that was most recently received from the WhatsApp server when the client last connected. This nonce will be changed by the server and sent every time a WhatsApp client connects to transact messages. And at the start of each new connection, which after all the clients initiate, the client will return the last nonce it received to the server.

Okay, clearly, if only one client is ever doing this for a given account, the server will always receive the nonce that it had previously sent to the client for that account. But if a user's WhatsApp identity authentication has been stolen and is being used by a physically separate client, then two clients will both be returning nonces, and only one of them will have the nonce that was most recently sent to that user's account by the server. As soon as the client that didn't most recently connect to the WhatsApp server does so, it will provide an obsolete nonce to the server, and the server will know that something's not right. So, yeah, it's a clean and simple solution to this cloned client problem. Again, not that big a deal, and certainly a good thing to do. Again, one wonders why this wasn't already in there, but it's good that it is now.

And lastly, this third feature they call Automatic Security Codes. They describe it this way. They said: "Our most security conscious users have always been able to take advantage of our security code verification feature, which helps ensure you're chatting with the intended recipient. You can check this manually by going to the encryption tab under a contact's info. To make this process easier and more accessible for everyone, we're rolling out a security feature based on a process called 'Key Transparency' that allows you to automatically verify that you have a secure connection. What it means for you is that when you click on the encryption tab, you'll be able to verify right away that your personal conversation is secured." Okay, now, once again, in kind of this touchy-feely boilerplate overview, they don't really explain what's going on, and they gloss over some of the, like, important stuff.

I read all the way through WhatsApp's description of this thing, what they're calling Automatic Security Codes. And you can thank me later for not dragging you through it. What it amounts to is a public, auditable, append-only - meaning nothing can ever be deleted; immutable is the other fun phrase for that - immutable log of the use of WhatsApp user account public keys. Now, the WhatsApp app already allows users to display, share, and verify their public keys with either a QR code or by verifying a 60, six zero, digit number. You know, which is each endpoint's public key.

But doing that requires that both parties arrange to interact in real time, preferably face to face, with their phones to exchange QR codes. And as we know, this was a feature that Threema, the secure messaging app, has used since its inception. In Threema you had like a little stoplight. You had green, yellow, and red. Or I guess better put it would be red, yellow, and green, where green is the highest level of verification which is only obtainable if each phone gets shown to the phone that you're peering with, and they're able to exchange their identities, like literally face to face.

Okay. So what WhatsApp is now creating under the name Key Transparency is a global public directory of user accounts and their matching public keys which will allow one-sided verification of the public keys of a user's contacts on WhatsApp. So, you know. And that's the key, and that's certainly useful. The point is that, in the encryption tab on

WhatsApp for one of your WhatsApp contacts, you could, if you clicked it, you got a QR code or a 60-digit number for that contact's public key. So you're seeing that. But in order to verify it, so all you know is you've got a public key; right? You don't know that it's actually theirs. In order to verify it, you both need to be available in real-time where your contact is able to read out their public key, and you verify that it matches the key that you believe is their public key. So thus a double-sided verification requirement.

What WhatsApp has done now is they're creating a global automation accessible, immutable append-only reference for all of the public keys of all of the WhatsApp users. So the users' experience is that now they will be able to press a button to verify the public key of any of their contacts. The contact gets looked up in this public key directory, the so-called Key Transparency Directory, and it's verified with the phone on a one-sided verification, not needing the other side. So it's another nice feature improvement, you know, in a way that makes sense. And, you know, good. I'm glad they have it. Again, WhatsApp is the number one global leader in secure messaging. By far and away they are in the lead, thanks to Facebook's prevalence. So it needs to be easy to use, and it needs to be as secure as possible.

Okay. The last thing I wanted to share before we get into the thing that terrified Google's Project Zero folks is a follow-up to a previous story. Remember back near the beginning of February of this year, it was Episode 909, Leo and I talked about a particularly astonishing and horrifying data breach which had occurred at a psychotherapy clinic in Europe. Here's how I described it then.

I said: "The news was that French authorities have detained a 25-year-old Finnish national who's accused of hacking the Vastaamo Psychotherapy Center. For reasons we'll see," I said, "this hack of Vastaamo is considered to be one of the worst in the country's history. Okay. Now, it occurred back in 2018 and 2019, so I guess this kid was, what, 20 years old then, when he allegedly stole the personal medical records of the clinic's patients and attempted to extort the clinic. To put pressure on the company, the hacker leaked extremely sensitive client files on the dark web. When that failed, he sent emails with ransom demands to more than 30,000 of the clinic's patients, asking them each for 200 euros and threatening to publish their medical records if they did not pay up." To which Leo replied: "Oh, boy."

And I continued, saying: "Uh-huh. Finnish authorities formally identified the hacker in October last year when they issued a European arrest warrant for his arrest, and they detained him last week." Okay. And I said: "Okay, so this is brazen and bad; right? The hacker obtained extremely sensitive personal medical information and chose to use it to extort both the clinic and its past patients, all 30,000 of them. And it was that number of files and patient histories that raised my eyebrows," I said, "30,000. Okay. No matter how large and busy this clinic might be, they cannot be currently treating 30,000 patients. And in fact there are 260 working days a year, five times 52. So if the clinic averaged 10 new patients per day, which seems like a high-side number, 30,000 patient records would be 11.5 years' worth of patient files at the rate of 10 per day."

Okay. So that was in early February. We have an update. For one thing, the Finnish psychotherapy clinic is now bankrupt. What a surprise! That's what'll happen when apparently the entire past hyper-confidential history of your clients is publicly exposed to the Internet. Who's going to make an appointment after that? So the clinic is gone.

What's a bit more interesting is that the ex-CEO of the clinic also faced criminal charges. And it's difficult to feel sorry for him since, as we know, mistakes can happen, right, by mistake; but policies do not happen by mistake. They happen by policy. So someone needed to be held accountable for this clinic's online preservation of the past records of 30,000 patients, which, you know, as I said back in February, if you really have to keep it for, like HIPAA-equivalent regulations, okay. But take it offline. Don't leave it on, you

know, exposable, accessible to the Internet, where it was. It turns out that this kid, this 25 year old, didn't hack into their network. He used a password in order to access the database over the Internet. So that's just unconscionable.

Okay. In the process of digging around within the facts of this data breach, it came to light that in addition to failing to take this sort of data security, like any sort of data security precautions that any medical patient would reasonably assume were in place, and that the law would both expect and require to have in place, the clinic's CEO knew about his company's sloppy cybersecurity for up to two years before the blackmail which took place in 2020.

It turns out that those earlier attacks that I mentioned in 2018 and 2019 were separate events that were previously known. They happened under this guy's watch. The clinic suffered several previous data breaches, as I said, in 2018 and 2019, and never reported them, presumably hoping that no traceable cybercrime would arise as a result, and thus the company would never be held to account. Of course, the final breach was what in the security industry we call "a doozy." So there was no sweeping that one under the rug. When you send email to 30,000 previous clients asking them for 200 euros each or their private psychotherapy records will be made public, that's not good.

Okay. Moreover, current breach disclosure and data protection regulations, such as our favorite GDPR in Europe, make it very clear that data breaches can no longer simply be ignored with the hope that no one will find out. No. They must be promptly disclosed for the greater good of all. So the final news from Finland is that our arguably criminally negligent CEO was, in fact, found by the courts to be criminally negligent and has been convicted and given a prison sentence. And one wonders whether he'll have the cell next to the young hacker who stole his data.

The hope is that this will serve to remind other business leaders that merely promising to look after other people's personal data is not good enough. You actually have to at least be able to demonstrate that you tried to do so. And that clearly did not happen in this case. And the fact that after multiple breaches, no improvements were made in the clinic's security posture. Nothing was done to fix those previous problems. So, yeah, it's good, as we know, that people are beginning to be held accountable because nothing will change otherwise.

Okay, Jason. One last mention of why we're here, and then we're going to look at ForcedEntry, the thing that the Project Zero guys at Google discovered that really did keep them awake at night.

JASON: That may surprise you, dot dot dot.

Steve: Terrified.

JASON: I keep going to that because apparently it works for clickbait. All right. We've done a lot of really big setup for this main event here. I'm super curious to know what they've found because it doesn't sound good.

Steve: It's not good. Okay. So, but it's way interesting, so that's of course the criteria for it making the podcast.

JASON: Yeah.

Steve: My original title for today's podcast was "KingsPawn." Now, that's capital P, not capital S, not KingSpawn, KingsPawn. That's the name that's been given to a powerful piece of spyware being offered by one of those Israeli spyware purveyors other than the NSO Group who, as we know, offer now their "too well known for comfort" Pegasus

spyware to various governments. But while doing the background legwork for that story, I ran across the fascinating technical details of another earlier piece of Spyware that was used by both this second group, who we will be talking about soon, and by NSO Group with Pegasus. I knew our listeners would find the technical details of this very interesting. So I decided to push our discussion of KingsPawn to next week, unless something even more juicy comes up. But we'll get to it.

Today, I want to take us through what Google's Project Zero team discovered about their so-called "ForcedEntry" exploit that has been successfully deployed by several malware vendors to gain entry into Apple iPhones. Citizen Lab, which is at the Munk School of Global Affairs and Public Policy at the University of Toronto in Canada, provided Google's Project Zero team with a sample of the ForcedEntry exploit; and Apple's Security Engineering and Architecture group collaborated with Google in working through exactly how it was possible for a GIF thumbnail image to take over any iPhone, despite all of Apple's many layers of protection which work to make that impossible. So what we had here with ForcedEntry was the holy grail of exploits providing attackers with remote code execution via a zero-click iMessage.

Citizen Lab somehow arranged to capture an NSO Group-generated, iMessage-based, zero-click exploit while it was being used to target a Saudi activist. At the time that Project Zero wrote about this, they said: "Based on our research and findings, we assess this to be one of the most technically sophisticated exploits we have ever seen, further demonstrating that the capabilities NSO provides rival those previously thought to be accessible only to a handful of nation states."

And of course what makes this so disturbing is that this is for sale, so that it's not only a handful of nation states that now have access to this sort of technology. Any qualified government - and some unqualified governments, unfortunately - can now purchase this advanced targeting malware. Okay. So here's how Project Zero described, just for reference, the NSO group. They wrote: "NSO Group is one of the highest profile providers of 'access-as-a-service,' selling packaged hacking solutions which enable nation-state actors without a homegrown offensive cyber capability to 'pay-to-play,' vastly expanding the number of nations with such cyber capabilities.

"For years," they wrote, "groups like Citizen Lab and Amnesty International have been tracking the use of NSO's mobile spyware package Pegasus. Despite NSO's claims that they 'evaluate the potential for adverse human rights impacts arising from the misuse of NSO products,' Pegasus has been linked to the hacking of the New York Times journalist Ben Hubbard by the Saudi regime, hacking of human rights defenders in Morocco and Bahrain, the targeting of Amnesty International staff, and dozens of other cases.

"The United States has added NSO to the 'Entity List' to severely restrict the ability of U.S. companies to do business with NSO, and stating in a press release that 'NSO's tools enabled foreign governments to conduct transnational repression, which is the practice of authoritarian governments targeting dissidents, journalists, and activists outside of their sovereign borders to silence dissent.'"

And they finished: "Citizen Lab was able to recover these Pegasus exploits from an iPhone, and therefore this analysis covers NSO's capabilities against iPhone. We are aware that NSO sells similar zero-click capabilities which target Android devices. Project Zero does not have samples of these exploits; but if you do, please reach out."

Okay. So in previous cases analyzed, targets were sent links, you know, like links in SMS messages. And while, as we know, these sorts of phishing-style attacks are often successful, individuals at the highest risk levels who are probably more technically savvy, or have received training about, like, where not to click, are less apt to click on something that doesn't really look absolutely 100% legitimate and pass a very stringent

smell test. So although a one-click exploit might work, the holy grail is the exploit that requires zero action on the part of the target. The exploit just works silently in the background.

Short of not using a device, there really is no way to prevent exploitation by a zero-click exploit. It's a weapon against which there is no defense. Since the initial entry point for Pegasus on iPhone using this attack, that is, this ForcedEntry attack, is iMessage, this means that a victim can be targeted just using their phone number or their Apple ID username, which is typically not difficult to find or obtain for high-profile individuals.

Okay. iMessage offers native support for GIF images which can be directly sent and received in iMessage chats. They show up in the chat window's chat log timeline. As Leo and his Discord denizens are all too aware, GIF images can also be animated by causing the GIF to contain a series of frames which are displayed in succession, creating animation. Apple wanted to make those GIFs, which don't naturally loop, loop endlessly, rather than play through only once.

So very early in the incoming iMessage parsing and processing pipeline, after a message has been received, but well before the message is displayed, iMessage calls a method in the IMTranscoderAgent process to which it passes any image file it receives with the extension .gif. The method called is named "IMGIFUtils," with "CopyFromPath" and "ToDestinationPath" parameters. As suggested by the selector's name, the intention was presumably to just copy the GIF file before editing the loop count field to make the GIF repeat endlessly. The method uses the Core Graphics APIs to render the source image to a new GIF file at the destination path. And although the source filename does need to end in .gif, that doesn't mean it's actually a GIF file.

So apparently we're going to learn this lesson all over again, or at least a variation of it. Apple's Image I/O library is used to determine the actual format of the source file, ignoring the file extension, and then parse it, completely and deliberately ignoring the file extension. This was actually done to prevent file type confusion attacks, which we've seen in the past. But in this instance, the protection backfired. Thanks to the use of this "fake gif" trick, over 20 different image processing codecs are now suddenly part of the iMessage zero-click attack surface.

Unfortunately, this includes some very obscure and complex formats. As a consequence of this unintentional chain of dependencies, hundreds of thousands of lines of code are now remotely exposed. Remember last week's podcast, "A Dangerous Interpretation." That was just one example of how difficult it can be to get complex codecs exactly right.

Okay. So what codec do you imagine the NSO Group chose to invoke with their "fake GIF in an iMessage" attack? Would you believe a PDF? Yes, believe it or not. The unintended chain of dependencies in iMessage's processing of incoming GIF images allowed external remote malicious parties to send anyone a PDF - but it's a PDF with a .gif extension; right? - which the receiving iPhone would then attempt to parse and display. For how many years were we, as an industry, dealing with malicious PDFs?

JASON: Yeah.

Steve: You know? Wow.

JASON: Lesson learned.

Steve: Oh, my god.

JASON: Or so we thought.

Steve: Yes, exactly. Wouldn't you think? So as we know, PDFs were one of the more popular exploitation targets due to the file format's ubiquity and the complexity of its - wait for it - its interpreter. And on top of the difficulty of rendering a document image from a typesetting specification, which is what PDFs are, PDFs later acquired the ability to interpret JavaScript, which is where we all intone in unison: "What could possibly go wrong?"

Okay. Fortunately, Apple's Core Graphics PDF parser does not appear to interpret JavaScript. So we can be thankful for that. But those evil geniuses - and as you'll soon see, I'm not using the term "genius" casually. I rarely use the term. The evil geniuses at the NSO Group found something sufficiently powerful lying inside the Core Graphics PDF parser.

Turning the clock way back to the late 1990s, we can all remember a time when bandwidth and storage were much more scarce commodities than they are today, and when it made much more sense to actually code in assembly language. I'm just saying. But back then, a compression standard known as JBIG2 was created. JBIG2 had a very narrow and very specific purpose, but it was very good at it. It was an image codec designed to highly compress monochrome images where pixels were only black and white, and where there was an extremely high degree of redundancy of what was being compressed. It was developed to achieve extremely high compression ratios for scans specifically of text documents, and was implemented and used in high-end office scanner and printer devices. If you were to use the direct "Scan to PDF" feature of such a machine, a PDF file would be produced which was mostly just a thin PDF wrapper around a JBIG2 compression stream.

As it happens, the PDFs produced by those scanners were exceptionally small, often being on the order of only a few kbytes. Okay. You might ask, how is this possible? There were two novel techniques which JBIG2 used to achieve its extreme compression ratios, which are relevant to the exploitation we're talking about here.

The first technique was known as Segmentation and Substitution. Okay. Segmentation as in looking at the individual black blobs on the page, meaning the text characters; and substitution as in let's not compress all of them. Let's just compress one of them and then point to the others. If you think about it, a text document contains a bunch of text. And in a language such as English, having a relatively small alphabet, the same character, like "s," "t," and "e," will appear all over the page. Believe it or not, JBIG2 actually segments documents to be compressed into their individual glyphs and uses a simple pattern recognition matching to match and collect all of the glyphs that look almost the same.

Then, rather than storing all of those near-duplicate glyphs, JBIG2 pretends that they are all the same and replaces all of them with just one. This replacement of all occurrences of similar-looking glyphs with a copy of just one often yields a document which remains entirely legible and enables very high compression ratios since the compressor just needs to store the coordinates of each of the same-looking glyphs rather than the glyphs themselves each time. The output remains perfectly readable, but the amount of information to be stored is significantly reduced.

Again, rather than needing to store all the original pixel information for the whole page, we only need one compressed version of the "reference glyph" for each character shape, and the relative coordinates of all of the places where its copies should be placed. This means that the decompression stage treats the output page like a canvas onto which it "paints" that one copy of the glyph at each of the stored locations.

There is one big problem with such a scheme. It's possible for a poor encoder to accidentally get confused and exchange similar-looking characters. This explains why the

format fell out of favor and became obscure. It does not explain, though, why Apple never thought to remove its decompression codec from their Image I/O library. Most of Apple's core PDF decoder appears to be Apple's own proprietary code; but the JBIG2 implementation is straight from Xpdf, whose source code is freely available. This means that the NSO Group effectively had the full source code for an extremely complex and not well vetted interpreter, to which they were able to provide any amount of data and cause to be invoked on any unsuspecting target's iPhone.

The vulnerability that was found in the code and exploited was a classic integer overflow, which occurs when the code processes a deliberately malformed document containing a JBIG2-compressed image description. This flaw allows the contents of the image to be executed as code, allowing the attackers total freedom to inject and run any code of their choosing. The only problem is that the messages sent remain on the target's device. And it was the persistence of the attacking code, that is, on this Saudi individual's phone, which they were able to provide to Citizen Lab, that triggered the unraveling of this entire scheme.

Okay, now, the extent to which the NSO Group went in their implementation of this vulnerability is somewhat astonishing because the rendering of this was sandboxed by - I can't remember the name of it, Blockade or something, that Apple has that sandboxes this. So they needed a sandbox escape. And doing that is what upped the ante for this entire attack.

Since replacing all of an image's text with identical duplicate versions of just one master reference character will result in an inherently lossy compression, right, because the decompressed image, that is, the reconstructed, essentially, image will not be identical to the original; right? Because one instance of "e," the character "e," has been used everywhere. So it's close, but it's not identical. So there was an additional mode added to JBIG2 to allow for lossless compression. An error mask would be created to represent the difference between the original image and the lossy super-compressed image. And it turned out that this error mask was not that large, either.

However, this capability, the capability of an error mask, gave the decompressor the ability to perform XOR operations since the mask was an XOR mask which was able to flip individual bits in the image as needed to make the final output identical to the original input. This meant that the decompressor's interpreter had access to AND, OR, XOR, and XNOR logical operators. And it turns out that, given those four operations, any computable function can be computed. It is, as we say, "Turing Complete."

JBIG2 does not itself have scripting capabilities. But when combined with a vulnerability which they found, it does have the ability to emulate circuits of arbitrary logic gates operating upon arbitrary memory. So these genius attackers must have thought, okay, why not just use that to build an entire computer architecture from scratch and script it? And believe it or not, this is exactly what the NSO Group's ForcedEntry exploit does.

By using over 70,000 JBIG2 interpreted segment commands defining logical bit operations, they defined, they essentially built an entire small computer architecture with features such as registers and a full 64-bit adder and comparator which they then use to search the iPhone's memory to perform arithmetic operations. It's not as fast as Javascript, but who cares? It's fundamentally computationally equivalent.

Okay. So here's what the Project Zero guys wrote to describe their amazement and their reactions to what they discovered once they had reverse engineered this exploit. They wrote: "The bootstrapping operations for the sandbox escape exploit are written to run on this logic circuit, and the whole thing runs in this weird, emulated environment created out of a single decompression pass through a JBIG2 stream. It's pretty incredible and, at the same time, pretty terrifying."

So from what I've just described, it's clear why the Project Zero guys, who are, you know, they're immersed in and live this sort of stuff day and night, they've seen it all, nevertheless they've never seen anything like this. They described this as the single most sophisticated attack they had ever encountered. On the one hand it amazed them, as it would anyone. And it also terrified them to think that there are people out there somewhere who are not only willing to go to these lengths, but who have the capability to do so. This is a tour de force in fundamental computer science. These people built a working emulation of a 64-bit computer, replete with registers and math capability, out of the pixels being manipulated by a JBIG2 image decompressor.

You know, and impressive as this is as an example of astonishing computational mastery, it's also a tragedy that the entire goal of this effort, the sole purpose was to subvert iPhones belonging to journalists and human rights defenders for the purpose of illegally spying on their actions and communications. It really seems a shame for such talent and capability to be spent toward such an end.

JASON: Yeah. John just whispered in my ear a very, you know, a truth that the purpose also was to sell the tools to people who wanted to do that. So it's finding something that is of value for others; right?

Steve: Good point. Very good point.

JASON: Yeah, that is fascinating. That is really, like, remarkable and, like they say, terrifying. Although every time I heard you say JBIG, I was reminded that was my football nickname back in high school, JBIG. Not really. I didn't play football. But it could have been. Yeah, that's pretty awesome and crazy stuff there, Steve.

Steve: Wow. So, yeah, to me, as someone who loves computing, I mean, this could be a Ph.D. thesis. You would be Dr. Abraham or whoever if you made this thing public. I sure hope that whoever did this got a lot of money themselves, I mean, evil as it is, and that they were appreciated for doing the impossible. And, you know, the thing I forgot to add is that also this makes you wonder. They would not have gone to this length if there was an easier way to skin this particular cat.

JASON: Yes, it's a good point.

Steve: Which suggests, my god, the bar has been raised by all of the security infrastructure that Apple has managed to cram into the iPhone.

JASON: If this is what you have to do to get there, then they've done a pretty darn good job. And, you know, as you've said many, many times on this show, there is no perfect security; right? Any system, I think, at some point there is some way in. It's just a matter of finding it. It turns out in this case you've got to go to great lengths. But it really is remarkable what they were able to accomplish to make that happen. Where there's a will, there's a way.

Steve: Wow.

JASON: Yeah, pretty incredible. Well, Steve, thank you so much for breaking that down, this and all the news. And every single week, week in, week out, all the security news, that's what Steve is great at, putting it into terms that you and I can understand because sometimes these stories, if I was to just read the article about this, I'd be lost. But thanks to Steve breaking it down, I have a better understanding of what's exactly going on here. And if you find yourself in the same position, and you want to support Steve and everything that's he's doing, go to GRC.com.