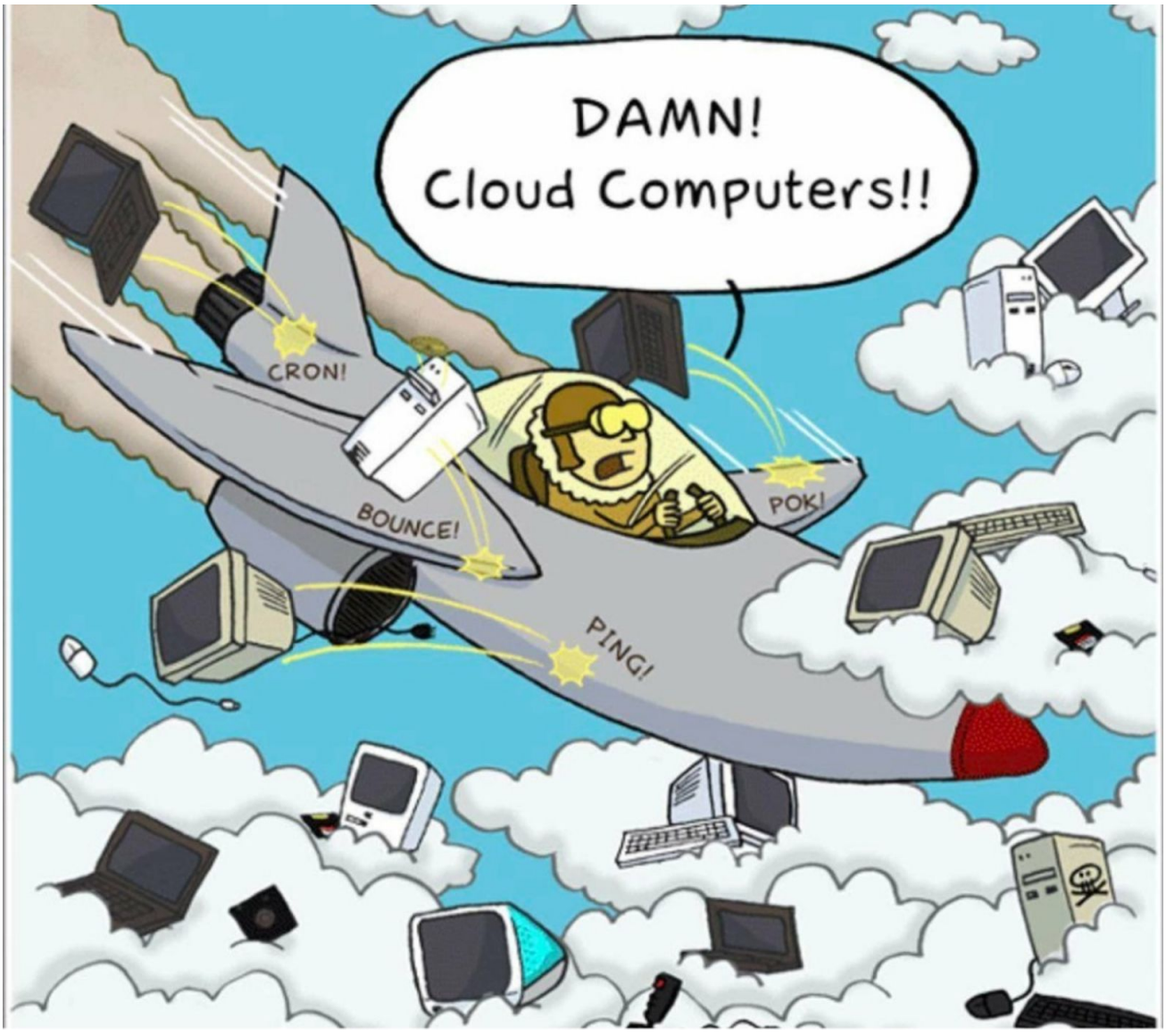
Security Now! #802 - 01-19-21

Where the Plaintext is

This week on Security Now!

This week we look at one aspect in which Chrome and Chromium differ and then at a bit of growth news from the DuckDuckGo folks. Google's Project Zero folks report on some terrific detective work, and we look at last week's Patch Tuesday. There's also Microsoft's pending change to the flaws which enabled last year's ZeroLogon debacle, and the NSA's interesting statement about enterprises and the DoH protocol. We look at the research that cracked the secret key out of Google's supposedly uncrackable Titan FIDO U2F dongle and we catch up with a bit of listener feedback. Then we wrap up by looking at various aspects of the frenzy caused by WhatsApp's quite predictable move to incorporate its user's conversation metadata into Facebook's monetization ecosystem.



Browser News

When is Chrome NOT Chromium?

So we all know that Google's world wide leading Chrome browser is based upon the open source and very solid Chromium core. This tends to beg the question: How are Chrome and Chromium similar and different?

At least one way in which Chrome is meant to be different from Chromium is in the many Google-specific features that are available only in Chrome. These are things like Chrome sync and Click to Call, or signing into the user's Google account and transacting personal sync data such as bookmarks, passwords, history, open tabs, settings, preferences, and, in some cases, even payment info saved in Google Pay.

The Chrome API provides direct hooks into those services that no other 3rd-party web browser is meant to have. Yet Google Chrome Engineering Director Jochen Eisinger explained that during a recent audit they discovered that some of the 3rd-party Chromium-based browsers were integrating features that are only intended for Google's use with Chrome. These APIs were intended to be private and specific only to Google Chrome. They were never intended to be integrated for any non-Google non-Chrome use.

Jochen declined to indicate which browsers had integrated Chrome Sync without authorization, but he did say that Google will block this unintended behavior starting on March 15th. By removing access to Chrome sync for other Chromium web browsers, it removes their ability to integrate the Chrome Sync API to sync their users' data to all devices where they're logged into their Google account.

However, Google did explain that users who have accessed private Google features such as Chrome Sync while using third-party browsers will still be able to access this synced data locally or in their Google account, depending on their sync settings. They will also be able to manage their data by going to the "My Google Activity" page, as well as downloading it from the Google Takeout page, and/or delete it by going here.

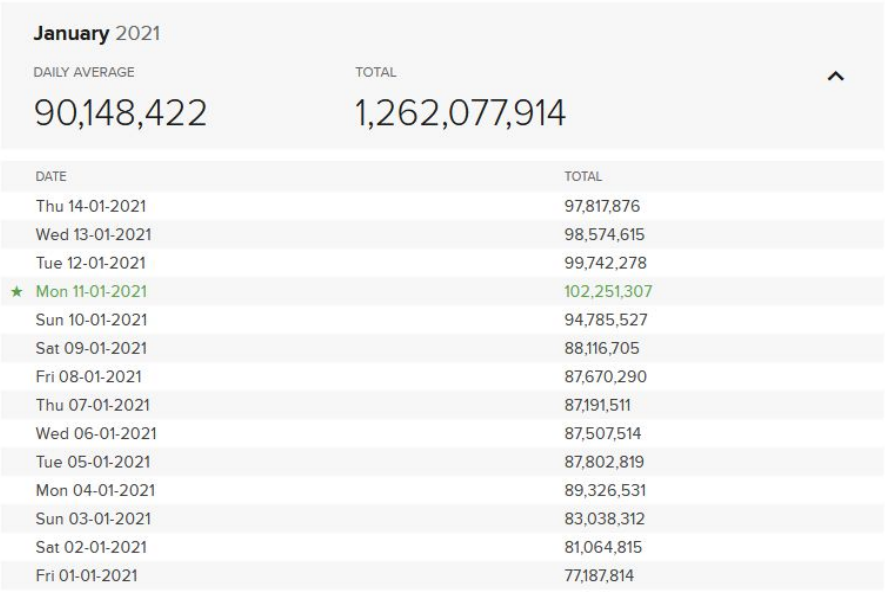
"Duck It!"

Though it's not a browser, this bit of news related to our browsing. Not long ago we talked about the privacy-centric DuckDuckGo! Search engine and we had some fun talking about children's games — after one of which "DuckDuckGo!" was unfortunately named. And we imagined that much "Google It!" has obtained generally understood meaning, that perhaps someday the same will happen with "Duck It!" — although I'm not holding my breath.

But, despite its name, DuckDuckGo! is a serious contender in the rarified search engine space. So I wanted to take a moment to share a milestone that it reached for the first time last Monday: Last Monday, for the first time in its 12-year history, it recorded its first-ever day of more than 100 million user search queries.

For the past two years DDG has been in a period of sustained growth. And especially since last August, when DDG began seeing more than 2 billion search queries a month on a regular basis. And while those numbers pale in comparison to search giant Google's 5 billion daily search

queries, DDG's slow but steady growth demonstrates that users really are looking for less invasive alternatives.



January 2021

DAILY AVERAGE	TOTAL
90,148,422	1,262,077,914

	DATE	TOTAL
	Thu 14-01-2021	97,817,876
	Wed 13-01-2021	98,574,615
	Tue 12-01-2021	99,742,278
★	Mon 11-01-2021	102,251,307
	Sun 10-01-2021	94,785,527
	Sat 09-01-2021	88,116,705
	Fri 08-01-2021	87,670,290
	Thu 07-01-2021	87,191,511
	Wed 06-01-2021	87,507,514
	Tue 05-01-2021	87,802,819
	Mon 04-01-2021	89,326,531
	Sun 03-01-2021	83,038,312
	Sat 02-01-2021	81,064,815
	Fri 01-01-2021	77,187,814

DuckDuckGo! has also expanded beyond its own browser URL by offering mobile apps for Android and iOS as well as a dedicated Chrome extension. And in a Tweet last September, DDG indicated that the apps and extension had been installed more than 4 million times. And in another win for the company, DuckDuckGo has been selected as the default search engine in the Tor Browser. And it will often appear as the default search engine in the private browsing modes of several other browsers.

Security News

Project Zero In the Wild

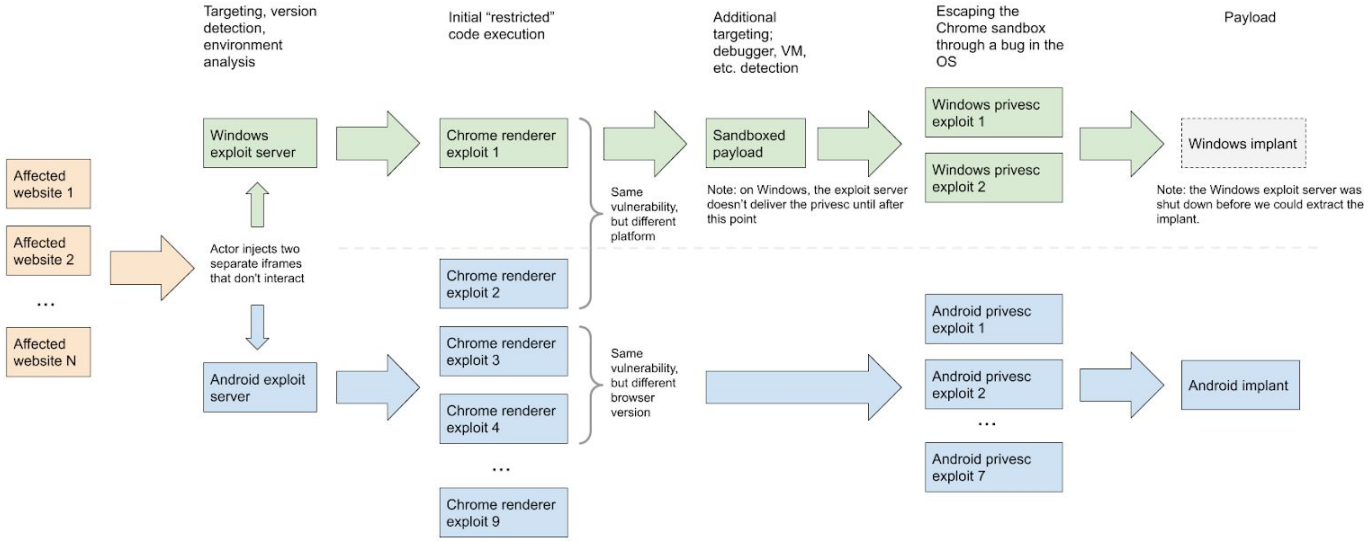
Last Tuesday Google's Project Zero published a six-part series which takes a deep analytical look at a discovery the group made during the 1st quarter of last year.

Last week I was talking about watering hole attacks where a browser vulnerability — it was that SCTP protocol vulnerability — where, unlike a JavaScript vulnerability, in order to be exploited the user would need to visit, often being lured to, a so-called "water hole server."

I especially appreciated the way Google framed their campaign. Here's how they began the first of size pages:

> At Project Zero we often refer to our goal simply as “make 0-day hard”. Members of the team approach this challenge mainly through the lens of offensive security research. And while we experiment a lot with new targets and methodologies in order to remain at the forefront of the field, it is important that the team doesn’t stray too far from the current state of the art. One of our efforts in this regard is the tracking of publicly known cases of zero-day vulnerabilities. We use this information to guide the research. Unfortunately, public 0-day reports rarely include captured exploits, which could provide invaluable insight into exploitation techniques and design decisions made by real-world attackers. In addition, we believe there to be a gap in the security community’s ability to detect 0-day exploits.
>
> Therefore, Project Zero has recently launched our own initiative aimed at researching new ways to detect 0-day exploits in the wild. Through partnering with the Google Threat Analysis Group (TAG), one of the first results of this initiative was the discovery of a watering hole attack in Q1 2020 performed by a highly sophisticated actor.

We discovered two exploit servers delivering different exploit chains via watering hole attacks. One server targeted Windows users, the other targeted Android. Both the Windows and the Android servers used Chrome exploits for the initial remote code execution. The exploits for Chrome and Windows included 0-days. For Android, the exploit chains used publicly known n-day exploits. [Referred to as "n-day" because, as I've often noted, they're only true 0-day's if they are NOT previously known.] Based on the actor's sophistication, we think it's likely that they had access to Android 0-days, but we didn't discover any in our analysis.



[Note that this STRONGLY implies the use of malvertising since this shows multiple affected websites each of which injects a pair of non-interacting iframes. One iframe pulls its content from the Windows exploit server whereas the other iframe pulls its content from the Android exploit server.]

From the exploit servers, we have extracted:

- Renderer exploits for four bugs in Chrome, one of which was still a 0-day at the time of the discovery.
- Two sandbox escape exploits abusing three 0-day vulnerabilities in Windows.
- A "privilege escalation kit" composed of publicly known n-day exploits for older versions of Android.

The four 0-days discovered in these chains have been fixed by the appropriate vendors.

We understand this attacker to be operating a complex targeting infrastructure, though it didn't seem to be used every time. In some cases, the attackers used an initial renderer exploit to develop detailed fingerprints of the users from inside the sandbox. In these cases, the attacker took a slower approach: sending back dozens of parameters from the end users device, before deciding whether or not to continue with further exploitation and use a sandbox escape. In other cases, the attacker would choose to fully exploit a system straight away (or not attempt any exploitation at all). In the time we had available before the servers were taken down, we were unable to determine what parameters determined the "fast" or "slow" exploitation paths.

The Project Zero team came together and spent many months analyzing in detail each part of the collected chains. What did we learn?

These exploit chains are designed for efficiency & flexibility through their modularity. They are well-engineered, complex code with a variety of novel exploitation methods, mature logging, sophisticated and calculated post-exploitation techniques, and high volumes of anti-analysis and targeting checks. We believe that teams of experts have designed and developed these exploit chains. We hope this blog post series provides others with an in-depth look at exploitation from a real world, mature, and presumably well-resourced actor.

The [remaining] posts in this series share the technical details of different portions of the exploit chain, largely focused on what our team found most interesting. We include:

- Detailed analysis of the vulnerabilities being exploited and each of the different exploit techniques,
- A deep look into the bug class of one of the Chrome exploits, and
- An in-depth teardown of the Android post-exploitation code.

In addition, we are posting root cause analyses for each of the four 0-days discovered as a part of these exploit chains.

Exploitation aside, the modularity of payloads, interchangeable exploitation chains, logging, targeting and maturity of this actor's operation set these apart. We hope that by sharing this information publicly, we are continuing to close the knowledge gap between private exploitation (what well resourced exploitation teams are doing in the real world) and what is publicly known.

https://googleprojectzero.blogspot.com/2021/01/introducing-in-wild-series.html

For anyone who wishes to follow-up and dig into the nitty-gritty's, the link to this page, which links to the others, in the show notes.

What I thought to be the single most important phrase in what I read was where the Porjet Zero guys wrote: "In addition, we are posting root cause analyses for each of the four 0-days discovered as a part of these exploit chains."

The key words there are "root cause analysis" — in other words: *"Oh, crap!! We just found another 0-day!! Always definitely better to have found it, than not. But MUCH BETTER for it never to have existed in the first place. So, rather than fixing it and saying "Whew!" and then going to lunch. Let's instead order-in and spend lunchtime asking ourselves how this fault was introduced in the first place, and what takeaway lessons we can learn to preemptively prevent anything like this from happening again."*

This is, I think, EXACTLY what needs to be done. And the continuing prevalence of software bugs which beleaguer our industry suggests that examining root causes occurs far too infrequently.

I've commented here on the podcast in the past that I truly love solving problems and puzzles with code. For me, it's a passion and a craft. So whenever I find a bug in my own code, I come to a dead stop and spend some time asking myself and exploring how that bug happened in the first place. What was I thinking? Exactly what did I get wrong? Is it likely to have happened elsewhere? Do I have a bad habit that needs correcting? In other words: Rather than being embarrassed about a bug, be informed by it. Some incorrect way of thinking caused it. I would submit that the only way for any craftsman to improve at their craft is to carefully examine and understand their mistakes.

1st Patch Tuesday of 2021

... Which leads us gracefully into our monthly review of Microsoft's Patch Tuesday, the first one of 2021.

Last Tuesday's updates fixed a total of 83 vulnerabilities, 10 of which Microsoft classified as CRITICAL and another was a 0-day which we'll get to in a moment.

Two of the CRITICAL vulnerabilities that were fixed were remote code execution vulnerabilities in Microsoft DTV-DVD Video Decoder and the HEVC Video Extensions (again). There was a remote code execution vulnerability in GDI+ and a critical memory corruption vulnerability in Microsoft Edge. The remaining five critical remote code execution vulnerabilities were all found and fixed in Microsoft's Remote Procedure Call (RPC) Runtime... Which is really not where you want to have RCE vulnerabilities (not that you want to have them anywhere.)

But the biggie was the 0-day that was found in Microsoft Defender being executed in the wild. It's being tracked as CVE-2021-1647 and it's a remote code execution flaw in Microsoft's Malware Protection Engine component (mpengine.dll)... and a proof-of-concept (PoC) exploit for the flaw is circulating publicly. We don't have any indication of how widespread the exploitation is, nor anything about its nature. But by now, one week post patch, I'd imagine that everyone has updated and rebooted. If not, it might be wise to do so since these fixes are a bit more critical, even if less numerous, than usual.

ZeroLogon Drop Dead — Next Patch Tuesday (Feb 9th)

As we know from its extensive abuse last year, the discovery and publication of the ZeroLogon flaw in Microsoft's previously-believed-to-be-more-secure-than-it-turned-out-to-be RPC Netlogon protocol, provided the perfect means for attackers to move laterally within an organization by effortlessly logging onto an enterprise's Master Domain Controller. This gave them the keys to the kingdom.

Microsoft's first of two patches, last August, fixed the security problem between Windows devices to reinforce truly secure Remote Procedure Call (RPC) communication for machine accounts on Windows devices, trust accounts, as well as all Windows and non-Windows Domain Controllers.

But non-Windows devices are also users of RPC, and many have never bothered to implement secure RPC. Therefore, since August, IT admins have been able to log all non-secure use of RPC by the various devices within their infrastructure in order to help them prepare for February's Drop Dead day.

After February 9th, Domain Controller enforcement mode will be enabled by default. Domain Controller enforcement mode requires that all Windows **and** non-Windows devices use secure RPC with Netlogon secure channel — unless customers have explicitly allowed the account to be vulnerable by adding an exception for the non-compliant device. And now that everyone knows how insecure NetLogon is without secure channel enabled, that's probably not something that anyone wants to do.

The NSA warns against outsourcing DoH services
https://media.defense.gov/2021/Jan/14/2002564889/-1/-1/0/CSI_ADOPTING_ENCRYPTED_DNS_U_OO_102904_21.PDF

In their 7-page advisory titled "Adopting Encrypted DNS in Enterprise Environments" last Thursday, the NSA warned against outsourcing DoH services to a 3rd-party provider. Their Executive Summary introduction explained their position:

> Use of the Internet relies on translating domain names (like "nsa.gov") to Internet Protocol addresses. This is the job of the Domain Name System (DNS). In the past, DNS lookups were generally unencrypted, since they have to be handled by the network to direct traffic to the right locations. DNS over Hypertext Transfer Protocol over Transport Layer Security (HTTPS), often referred to as DNS over HTTPS (DoH), encrypts DNS requests by using HTTPS to provide privacy, integrity, and "last mile" source authentication with a client's DNS resolver. It is useful to prevent eavesdropping and manipulation of DNS traffic. While DoH can help protect the privacy of DNS requests and the integrity of responses, enterprises that use DoH will lose some of the control needed to govern DNS usage within their networks unless they allow only their chosen DoH resolver to be used. Enterprise DNS controls can prevent numerous threat techniques used by cyber threat actors for initial access, command and control, and exfiltration.
>
> Using DoH with external resolvers can be good for home or mobile users and networks that do not use DNS security controls. For enterprise networks, however, NSA recommends using only designated enterprise DNS resolvers in order to properly leverage essential enterprise cybersecurity defenses, facilitate access to local network resources, and protect internal network information. The enterprise DNS resolver may be either an enterprise-operated DNS server or an externally hosted service. Either way, the enterprise resolver should support encrypted DNS requests, such as DoH, for local privacy and integrity protections, but all other encrypted DNS resolvers should be disabled and blocked. However, if the enterprise DNS resolver does not support DoH, the enterprise DNS resolver should still be used and all encrypted DNS should be disabled and blocked until encrypted DNS capabilities can be fully integrated into the enterprise DNS infrastructure.
>
> This guidance explains the purpose behind the DoH design and the importance of configuring enterprise networks appropriately to add benefits to, but not hinder, their DNS security controls. The following recommendations will assist enterprise network owners and administrators to balance DNS privacy and governance.

And, for what it's worth, I completely agree with every sentence in that Executive Summary.

I think that they got it all exactly right.

The only issue I might take is wondering about the value, let alone the necessity, of enabling DoH within the enterprise at all. I don't see what value it provides. Tech and privacy-minded employees might want to sneak their browser traffic out of the organization without being monitored. But we're always reminded that the use of employer networks is the employer's to oversee and regulate. So I would think that enterprises would be entirely correct to block the use of DoH and require browsers to use standard DNS and the organization's DNS with whatever added security filtering and malware detection the enterprise wishes to deploy.

The real value of DoH is ISP and access-point spying, neither of which are present within an enterprise network.

A Side-Channel in Titan
https://ninjalab.io/a-side-journey-to-titan/

The guys at NinjaLab performed a classic side-channel attack on a Google Titan FIDO U2F security key and by watching it work while monitoring its electromagnetic (RF) radiations, they successfully extracted its embedded super-secret, deliberately designed to never be extracted, private key.

We've talked about side-channel attacks a lot in the past. Sometimes the power drawn by a device while it's performing cryptographic operations will fluctuate. Sometimes the power supply will emit different sounds. Or perhaps the timing required to perform an operation might vary. It could be anything. The coolest way of stating the goal of avoiding any possibility for a side-channel is that nothing about the behavior of a device should vary as a function of any internal secret bits. So, taking timing as an example, the time required to perform some operation, typically the pattern of jumps taken in the code, should be completely independent of the secrets being kept. So, the bits of the secret key must not cause the algorithm's code path to vary.

Some things are feasible to hold constant. But others fall below one's ability to control... such as the instantaneous electromagnetic radiation of an integrated circuit that's doing the work.

The Google Titan Security Key is a FIDO U2F hardware device proposed by Google (available since July 2018) as a two-factor authentication token to sign in to applications (e.g. your Google account). Our work describes a side-channel attack that targets the Google Titan Security Key's secure element (the NXP A700X chip) by the observation of its local electromagnetic radiations during ECDSA signatures (the core cryptographic operation of the FIDO U2F protocol). In other words, an attacker can create a clone of a legitimate Google Titan Security Key.

To understand the NXP ECDSA implementation, find a vulnerability and design a key-recovery attack, we had to make a quick stop on Rhea (NXP J3D081 JavaCard smartcard). Freely available on the web, this product looks very much like the NXP A700X chip and uses the same cryptographic library. Rhea, as an open JavaCard platform, gives us more control to study the ECDSA engine.

We could then show that the electromagnetic side-channel signal bears partial information about the ECDSA ephemeral key. The sensitive information is recovered with a non-supervised machine learning method and plugged into a customized lattice-based attack scheme.

Finally, 4000 ECDSA observations were enough to recover the (known) secret key on Rhea and validate our attack process. It was then applied on the Google Titan Security Key with success (this time by using 6000 observations) as we were able to extract the long term ECDSA private key linked to a FIDO U2F account created for the experiment.

Cautionary Note

Two-factor authentication tokens (like FIDO U2F hardware devices) primary goal is to fight phishing attacks. Our attack requires physical access to the Google Titan Security Key, expensive equipment, custom software, and technical skills.

Thus, as far as our study goes, it is still safer to use your Google Titan Security Key or other impacted products as FIDO U2F two-factor authentication token to sign in to applications rather than not using one.

Nevertheless, this work shows that the Google Titan Security Key (and other impacted products) would not avoid unnoticed security breach by attackers willing to put enough effort into it. Users that face such a threat should probably switch to other FIDO U2F hardware security keys, where no vulnerability has yet been discovered.

Some secrets cannot be kept, and the fancier the system is that tries, the more likely it is to have some behavior that gives it away.

Unlike a TOTP which uses symmetric cryptography where each end needs to share a private key, systems like FIDO and SQRL use asymmetric public key cryptography where the user holds their private key and the remote end holds their public key. This significantly reduces the risk to the user and to the system since the public keys need not be kept secret.

The takeaway for this hardware token hack is not to be too glib about the idea that the hardware token's private key cannot possibly be extracted. It clearly can be if someone is sufficiently determined.

Closing the Loop

Robert Rosenberg / @Bob_Rosenberg1

Several Podcasts ago, you were discussing the "PayPal Football". Well, I did it! I took it apart (ya gotta be careful, but it's not hard), found the battery type, bought one, and put it in. Next time I dealt with PayPal, I used the now-working Football, and PayPal took it! They authenticated on my dead-for-years token. I'm guessing it's a press-based OTP, not a time-based OTP.

[I completely agree with Robert, and I think that we must have known that once, but I forgot,

since I do recall nothing that our listeners discovered that one of the digits was a simple modulus 10 counter.]

Peter Sinclair / @peterwjsinclair
Re: "Out With the Old" — Great show Steve took me right back to the '80s. I agree with you stripping out the old code from Spinrite 6.1; we still have our 6.0 or older copies of if we want to run it on any heritage drives we might have lying around.

[I'm so glad Peter noted that, since I had forgotten to explicitly mention that all owners of SpinRite will continue to have access to previous versions 5 and 6. Just yesterday I removed support for diskette drives from the v6.1 code. Again, since they are so different from other mass storage drives, there was a bunch of explicit special case handling support laced throughout the code. And it's doing almost no one any good. But, diskettes ***are*** *a medium that can benefit significantly from SpinRite. And every so often, someone — typically an archaeologist — will come upon a diskette that's no longer readable but that still contains important data. In such cases we suggest that they download SpinRite v5 because for some reason v5 appears to do a better job on diskettes than v6. This has and still does bug me. I've stared at the code for hours and I cannot explain it, even though it appears to be true. So... use SpinRite v5 for diskettes. And you'll be reminded of this since SpinRite v6.1 won't even see the system's diskette drive, if, for some reason, there happen to be any.]*

oaksong @oaksong
@SGgrc glad to know I can still use those OnTrack managed drives.

[Ha! We know he was joking — at least I hope he was — but here again, SpinRite v6 still contains, and always will, all of that code for dealing with those long gone and nearly forgotten messy solutions such as OnTrack.]

Where the Plaintext is

WhatsApp has shown us that it's not only Zoom's CEO, Eric Yuan, who is deftly able to somehow locate the exact center of a large pile of poo to step in. In Eric's case, well meaning though he was, every time he opened his mouth in a well-meaning attempt to clarify something, he made things worse. Often much worse.

And now, over the past week, in an apparent gross misunderstanding of why their own users were using their secure messaging product in the first place, WhatsApp has managed to trigger such an exodus from their platform that the two primary alternatives, Telegram and Signal, have never seen such increases in their user base, and Signal has continued having trouble staying online amid the deluge.

We reported last week that Signal was struggling to authenticate the torrent of incoming users

since they require an SMS messaging loop and the cellular carriers were apparently having individual scaling issues. Then, that trouble apparently migrated from authentication to the service itself. This triggered Signal outages and a tweet last Friday from Edward Snowden:

> For those wondering about @SignalApp's scaling, #WhatsApp's decision to sell out its users to @Facebook has led to what is probably the biggest digital migration to a more secure messenger we've ever seen. Hang in there while the Signal team catches up. https://t.co/xNKz4aYUFJ
>
> — Edward Snowden (@Snowden) January 15, 2021

And the rush may have been further accelerated when Elon Musk simply tweeted the two words "Use Signal" to his 42.6 million followers.

One of the questions that inevitably arises from any use of a free service is "Why is it free?" and "Who is paying for this?" This, of course, reminds us of the famous joke: "Well, yeah, we do lose money on each one. But don't worry, we make it up in volume." This is one of the reasons why, if I **were** to use any 3rd-party messaging app (I don't, because I have no need for any.) I would still choose Threema, which is paid for one time to generate operating revenue for the company. And their "Threema Work" product for the enterprise uses a subscription model. So, again, we understand how they are monetizing.

Just as a reminder, WikiPedia has a short and recently updated history of Threema:

> Threema was founded in December 2012 by Manuel Kasper. The company was initially called Kasper Systems GmbH. Martin Blatter and Silvan Engeler were later recruited to develop an Android application that was released in early 2013.
>
> In Summer 2013, the Snowden leaks helped create an interest in Threema, boosting the user numbers to the hundreds of thousands. When Facebook took over Whatsapp in February 2014, Threema got 200,000 new users, doubling its userbase in 24 hours. Around 80% percent of those new users came from Germany. By March 2014 Threema had 1.2 million users. In Spring 2014, operations have been transferred to the newly created Threema GmbH.
>
> In December 2014, Apple listed Threema as the most-sold app of 2014 at the German App Store.
>
> In 2020, Threema expanded with video calls, plans to make its codebase fully open-source as well as introduce reproducible builds and Threema Education, a variation of Threema intended for education institutions.
>
> During the second week of 2021 *[that is to say, last week]*, Threema saw a quadrupling of daily downloads spurred on by controversial privacy changes in the WhatsApp messaging service. A spokesperson for the company also confirmed that Threema had risen to the top of the charts for paid applications in Germany, Switzerland, and Austria. This trend continued into the third week of the year, with the head of Marketing & Sales confirming that downloads had increased to ten times the regular amount, leading to "hundreds of thousands of new users

each day".

So, yeah, if I had any need for private messaging, Threema would be my choice. I would carefully and manually exchange my Threema public key with the people I need to converse with secretly, and that would be that.

When I was working with Rasmus on the SQRL / XenForo integration at his suggestion we used Signal, not because we needed any super secrecy, but because it had a convenient Windows desktop client and we were in different time zones. So we were able to easily exchange messages, screen shots and links that way. Was it comforting to know that our exchanges could not be intercepted and decrypted? Sure. That's what we expect these days, right?

But I'm interested in the massive effect that was driven by WhatsApp's statement that they intend to share messaging metadata with Facebook's other businesses. It's as if everyone using WhatsApp believed that Facebook purchased WhatsApp out of the goodness of their heart in order to offer the entire world a free messaging service without any strings. As we noted above about the influx to Threema when Facebook announced their purchase of WhatsApp, even back then not everyone believed in the free lunch theory. So when Facebook dropped their other shoe about their plans for further monetizing their WhatApp users, no one should have been surprised. It has always been clear that sooner or later Facebook would do this. And if they cannot peer into the conversations, then they'll at least collect everything they're able to from the outside. It seems to me that knowing who I'm talking to is an intrusion, but hey!... it's free!

The problem was that they made too large a change all at once. It's like the old joke about how to boil a frog. You don't toss a frog into boiling water because it will jump right out. Instead, you place the frog into cold and comfortable water then slowly turn up the heat. In retrospect, Facebook should have patiently and very slowly chipped way at the privacy of their WhatsApp users, through a series of much smaller privacy incursions. There is a very non-zero switching cost associated with changing messaging platforms. Right now, Twitter is full of people complaining that they just switched to Signal but no one they know is there. So it's not enough to make the move. You need to move your entire community. And that's difficult in the extreme.

So if Facebook had, instead, chipped away at their user's privacy, at each step the user would reconsider the switching cost of leaving WhatsApp — where all of their friends already are — and they might recall their disappointment the last time they tried, versus the loss of just one additional tiny little aspect of privacy that probably really doesn't matter that much anyway. If that had been the strategy it seems clear that many fewer would have left.

But as we know, instead of understanding this, WhatsApp and Facebook got a little too big for their britches and lowered the boom on their users all at once: *"Agree to this or else you're disconnected! After all, who would possibly want to leave WhatsApp? Oh my god it's such a wonderful thing! And besides, we're doing so much more than just metadata collecting everywhere else. So what's the big deal?"* Yikes! No wonder that the result has been a mass exodus.

The truth is, essentially no users understand the underlying technology and privacy guarantees

of the messaging systems they're using. So the illusion is enough. They want the illusion of complete privacy, even if they don't actually have much. Just tell them they do. Tell them it's good. Say that it's military grade encryption. Tell them how many billions of years it would take the world's largest supercomputer to crack their grocery list. But, whatever you do, don't mention that a simple keystroke logger would render every bit of that fancy-ass technology completely superfluous. They don't want to hear that.

Out of curiosity, I Googled the phrase "secure messaging apps" and I received a list of links:

- 10 Most Secure Messaging Apps - Best Encrypted Chat App
- Secure Messaging Apps Comparison
- The Best Encrypted Messaging Apps You Should Use Today
- Best encrypted messaging apps | Tom's Guide
- The Most Secure Messaging Apps for Android & iOS 2020 | AVG
- The Best Private and Secure Messaging Apps in 2021 | ExpressVPN

Does any of this matter? No. As we know, there ARE subtle differences between apps and their underlying technologies. Through the years we've covered them all, right here. Mostly, the differences boil down to the way the various apps exchange and manage their user's keys. Since the keys are everything. But here's the bottom line:

ANY proper use of encryption turns the conversation on the wire into maximum entropy noise with no discernible patterns and no feasible means of decrypting what's intercepted over the wire — ever. Yes, Virginia, your grocery list is safe!

But, that is the ONLY PROTECTION that ANY of these apps can provide. That's it. Period. The ONLY THING that any of these apps can guarantee is that **while the data is in motion** between endpoints it is utterly infeasible for it to ever be decrypted.

That's the only guarantee. Any keystroke logger defeats ANY of these apps no matter how many layers of military grade encryption have been employed after the keystrokes have been logged.

And, of course I'm using the term "keystroke logger" because it's such a clear and well-understood term. It stands-in for the concept that I want to convey, which is that on-the-wire encryption **no longer means anything** about our actual, true privacy. If you look at the packets moving across today's Internet, unlike the Internet of 15 years ago when we began this podcast, all you will ever see today is packets full of noise. Just 100% random bits. Today, it's all noise — it's all encrypted. And **no one** bothers looking at it, because anyone who wants to know what's in those packets also knows quite well that the encryption problem has long since been solved and none of that noise on the wire **can** be decrypted.

The apocryphal story of the infamous bank robber, Willie Sutton, applies here. The story goes that a reporter once asked Willie why it was that he robbed banks. And Willie was said to reply: "Because that's where the money is."

Updating the story to 2021, a non-technical supervisor at the NSA might ask one of their top technical managers why all of the Agency's work has been diverted to investments in keystroke logging? To which the technical manager would reply: "Because that's where the plaintext is."

The point I hope I have made is that, barring some implementation mistake, none of this "what's the best messaging app" conversation matters at all anymore. The real world has moved on. We've long observed that the best camera is the one you have with you. Similarly, the best messaging app is the one shared by the people with whom you wish to communicate. Period.

All other things being equal, I would choose one with a clear and simple economic model that I understand and that makes sense to me. If I don't care about my metadata being monetized, about Facebook tracking everyone with whom I communicate, then WhatsApp, whose economic model clearly requires it to eventually be snooping on me would be just fine. But if I do wonder who's paying for this service, and I would prefer for it to be very clearly me in exchange for not having the fact of my communications being monetized, then Threema would be my choice.

But in any event, the privacy provided by encryption should no longer factor into the equation. Because it doesn't matter at all. Anyone who wishes to have access to the content of our conversations will go after the plaintext.

I remain uncomfortable with the autonomous key management offered by many of these systems, including iMessage. But that’s also splitting hairs and doesn’t really matter either. If I really cared to have a private conversation I would never use a smartphone.