Security Now! #967 - 03-26-24

GoFetch

This week on Security Now!

After I comment on US Departement of Justice's antitrust suit against Apple, we'll update on General Motor's violation of its car owner's privacy and answer some questions, including what happy news is Super Sushi Samurai celebrating? Has Apple abandoned its plans for HomeKit-compatible routers? And what appears to be shaping up to take their place? Will our private networks be receiving their own domain names? And if so, what? The UN has spoken out about AI -- does anyone care? and what do I think the prospects are of us controlling AI? What significant European country just blocked Telegram? What did the just-finished 2024 Pwn2Own competition teach? Might the US be hacking back against China as they are against us? And after a bit of interesting SpinRite news and a bit of feedback from our listeners, we're going to spent the rest of our time looking into last week's quite explosive headlines about the apparently horrific unfixable flaws in Apple's M-series silicon. Just how bad is it?

A photo that's just BEGGING for the caption...

"What could possibly go wrong?"

Security News

Apple vs U.S. DoJ

Last Thursday March 21st was a rough day for Apple. Not only did the tech press explode with truly "hair on fire" headlines about critical, unfixable, unpatchable, deeply rooted cryptographic flaws rendering Apple's recent M-series ARM-based silicon incapable of performing secure cryptographic operations – which is the topic we'll be spending the end of today's podcast looking at in some detail, since it is ***super*** interesting – but also last Thursday, the United States Department of Justice was joined by 15 other states of our Union and the District of Columbia, in a lawsuit alleging that Apple has been willfully & deliberately violating Section 2 of the Sherman Antitrust Act. Five sentences from the DoJ's comments delivered last Thursday, read:

> *As our complaint alleges, Apple has maintained monopoly power in the smartphone market not simply by staying ahead of the competition on the merits, but by violating federal antitrust law. Consumers should not have to pay higher prices because companies break the law. We allege that Apple has employed a strategy that relies on exclusionary, anticompetitive conduct that hurts both consumers and developers. For consumers, that has meant fewer choices; higher prices and fees; lower quality smartphones, apps, and accessories; and less innovation from Apple and its competitors. For developers, that has meant being forced to play by rules that insulate Apple from competition.*

This is not a podcast about antitrust law. We all know I'm not an attorney nor am I trained in the law. So I have no specific legal opinion to render here. However, I've been a successful small business founder, owner and operator throughout my entire life. And I'm certainly a big fan and believer in the free enterprise system and in the principles of capitalism. But I also appreciate that this system of competition is inherently unstable. It has a natural tendency for the big to get bigger through acquisition and the application of economies of scale and leverage. That same system that creates an environment which promotes fair competition can be abused once sufficient power has been acquired.

Those of us of a certain age have watched Apple be born, then fall, only to then rise again from the ashes. My own first commercial success was the design, development, production and sales of a high-speed high-resolution light pen for the Apple II which allowed its users to interact directly with the Apple II screen. To my mind there is no question that as a society we are all richer for the influence that Apple's aggressive pursuit of perfection has had on the world. Things as simple as product packaging will never be the same.

But for some time we've been hearing complaints about Apple's having taken this too far. It's understandable for competitors to complain, and to ask the government to step in and do something. At some point that becomes the government's very necessary role, just as we saw previously when the same thing happened with Microsoft. For many years the US Government has done nothing, while Apple has continued to grow and continued to aggressively use its market power to increase its shareholder's wealth. The question is, when does **use** of market power become **abuse** of market power? The next few years will be spent in endless depositions and expert testimony working to decide exactly what sort of cage Apple needs to be constrained within.

One thing we know is that **many** of the arguments Apple will be making on its own behalf will involve security. The security inherent in its closed messaging system. The inherent security of its closed App Store. Apple will allege that by keeping its systems closed it is protecting its users from unseen nefarious forces. But, for example, the presence of Signal and WhatsApp in the Apple Store and on Apple devices, which create freely interoperable super-secure cross-platform messaging, suggests that Apple's own messaging technology could work similarly if they wished it to. During the news coverage of this since Thursday, I've encountered snippets of evidence which suggests that the government has obtained clear proof of Apple's true motives where Apple's technology has been designed to support Apple's interests rather than those of its users.

In any event, nothing is going to happen on this front for a long time. Years will pass, and this podcast will be well into four digits by the time anything is resolved with the DoJ's antitrust lawsuit. The way things have been going, it seems much more likely that the laws being written and enacted within the European Union today will be forcing Apple's hand long before the DoJ finishes making its case. All that may eventually be required will be for the U.S. to force Apple to do the same thing that they've already been forced to do throughout Europe.

But, as for whether Apple-designed silicon cannot perform secure cryptographic operations, **that** is something this podcast **can** speak to authoritatively, and we'll be doing so once we've caught up with other more interesting news and feedback!

G.M.'s Unbelievably Horrible Driver Data Sharing Ends

Last week we shared the hard to believe but true story that General Motors had been sharing (and by sharing I'm pretty sure the proper term would be selling) the detailed driving record data, with driving detail down to how rapidly its owner's car accelerated, how hard it braked and its average speed, with 3rd party data brokers who in turn — surprise! — resold that data to insurance companies. Happily, the outcry over this immediately changed GM's policy. Likely that, and the threats of lawsuits they must have received. ArsTechnica reported under the headline "GM stops sharing driver data with brokers amid backlash. Customers, wittingly or not, had their driving data shared with insurers.":

> *After public outcry, General Motors has decided to stop sharing driving data from its connected cars with data brokers. Last week, news broke that customers enrolled in GM's OnStar Smart Driver app have had their data shared with LexisNexis and Verisk.*
>
> *Those data brokers in turn shared the information with insurance companies, resulting in some drivers finding it much harder or more expensive to obtain insurance. To make matters much worse, customers allege they never signed up for OnStar Smart Driver in the first place, claiming the choice was made for them by salespeople during the car-buying process.*
>
> *Now, in what feels like an all-too-rare win for privacy in the 21st century, that data-sharing deal is no more. GM told Ars in a statement: "As of March 20th, OnStar Smart Driver customer data is no longer being shared with LexisNexis or Verisk. Customer trust is a priority for us, and we are actively evaluating our privacy processes and policies."*

I'll just bet you are!

Super Sushi Samurai

I saw this bit of happy cryptocurrency news that made me smile: It seems that last week the blockchain game, Super Sushi Samurai had $4.6 million worth of its tokens stolen. However, it just reported that they had all been recovered. What happened? They explained that the hack was actually the work of a security researcher who exploited a bug in their code to move the funds out of harm's way to prevent a future theft. Super Sushi Samurai described the incident as a "white hat rescue" and has ended up hiring the white-hat as a technical advisor. That's what I'd call a "G-rated" happy ending!

Apple has effectively abandoned HomeKit Secure Routers

Apple Insider had some interesting coverage about Apple's apparently failed initiative to move their HomeKit technology up into home routers. I was a fan of this, since it promised to provide router-enforced inter-device traffic isolation and the only place that can really be accomplished is at the router. Our listeners know that I've been advocating for the creation of isolated networks so that IoT devices would be kept separate from the household's PCs. But what Apple proposed five year ago back in 2019 would have additionally isolated each IoT device from all others. Here's what Apple Insider explained:

> *Apple's HomeKit Secure Routers were announced in 2019 but were never really taken up by manufacturers, and now some vendors are claiming Apple is no longer pursuing the technology. HomeKit Secure Routers were introduced by Craig Federighi at WWDC 2019 and in the same breath as HomeKit Secure Video. The latter took time to reach the market, but it was used, and many manufacturers adopted it, even if others would not.*
>
> *During CES 2024 [this year], two router vendors separately told AppleInsider that Apple is no longer accepting new routers into its program. If that claim is correct — and it probably is, since it came from the same rejected manufacturers — given the lack of HomeKit Secure Routers on the market, it appears that Apple has abandoned the idea even though Apple still has active support pages on the matter. But, it still has support pages on AirPort routers too, and those are as dead as a doornail.*
>
> *The idea was that (quoting) "with HomeKit at the router, we'll automatically firewall-off each of your accessories. So even if one were to be compromised, it wouldn't be able to access your other devices."*
>
> *Back in 2019, Federighi said: "Many accessories don't just connect via HomeKit, they also connect via the Internet and through your router, and unfortunately, this can leave them open to attack. We want to make sure that this can't happen. And so we're bringing HomeKit to routers."*

Except that, as we know, that didn't really happen.

> *During the 2019 announcement, Apple said that "the first HomeKit-enabled routers will be coming from Linksys, eero, and internet service providers like Charter Spectrum." Even if these unrelated vendors were both mistaken, and Apple has **not** abandoned HomeKit Secure Routers, it effectively has. More than four years later, Apple currently lists two HomeKit Routers on its site. They are the Linksys Velop AX4200 and the AmpliFi Alien.*

Eero has a notice saying that its Eero Pro 6E and Eero 6+ do not support Apple Home Kit and that they have no plans to offer Apple Home Kit's router functionality on those devices.

Linksys has ignored requests for comment. An eero spokesperson did offer a comment, but didn't answer AppleInsider's question. Saying: "HomeKit devices can connect to eero over Wi-Fi and Thread but we do not offer Apple's HomeKit router functionality on our latest devices" So while eero did not say why it has dropped HomeKit Secure Routers, there doesn't appear to be any plan to reintroduce them.

Of Apple's two remaining HomeKit Secure Routers listed on its site, the Linksys Velop AX4200 and the AmpliFi Alien, AmpliFi's own description doesn't mention HomeKit. The current Linksys listing does state that it works with HomeKit, but doesn't mention HomeKit Secure.

So that's one of Apple's 2019 launch partners gone and another reduced to a single under-promoted product. Even at that 2019 launch, the Charter Spectrum partnership seemed a bit of a stretch, and today it's hard to prove it ever released a product.

So the bottom line is that not everything that's announced happens. And asking router manufacturers to modify their firmware to incorporate the required Homekit functionality – and it appears that it may have taken some significant customization – was just never going to get off the ground.

And this is probably for the better since it appears that we have already (and blessedly quickly) moved beyond disparate proprietary closed IoT ecosystems. All the buzz appears to be surrounding the interoperability technology known as "Matter". This was formerly known as CHIP which stood for Connected Home over IP. It's now been rebranded MATTER and everyone appears to be seeing the light and no one wants to be left out. This includes Amazon with Alexa, Apple, yes, with HomeKit, Google's Home and Samsung's SmartThings. The Matter specification, now at v1.2 is open, open source and license free. Anyone can create Matter compatible devices which, if they follow the specification, will interoperate. With more than 550 companies having announced their commitment to Matter, it appears that it is rapidly growing to become the standard that everyone will need to follow.

Against that background, it's hardly surprising that Apple has apparently decided not to try to push their proprietary HomeKit technology onto router manufacturers.

The forthcoming ".INTERNAL" TLD

In a cool bit of news, ICANN, the Internet Corporation for Assigned Names and Numbers, is in the process of designating and reserving a top level domain specifically for use on private internal networks. In other words, our 10.Dot and 192.168.Dot networks will be obtaining an official TLD of their own – so "localhost" may soon be less lonely. Here's the Executive Summary which explains and lays out the rationale behind ICANN's plans:

In this document, the SSAC (ICANN Security and Stability Advisory Committee) recommends the reservation of a DNS label that does not (and cannot) correspond to any current or future delegation from the root zone of the global DNS. This label can then serve as the top-level domain name (TLD) of a privately resolvable namespace that will not collide with the resolution

*of names delegated from the root zone. In order for this to work properly, this reserved private-use TLD must **never** be delegated in the global DNS root.*

Currently, many enterprises and device vendors make ad hoc use of TLDs that are not present in the root zone when they intend the name for private use only. This usage is uncoordinated and can cause harm to Internet users.

The DNS has no explicit provision for internally-scoped names, and current advice is for the vendors or service providers to use a sub-domain of a public domain name for internal, or private use. Using sub-domains of registered public domain names is still the best practice to name internal resources. The SSAC concurs with this best practice, and encourages enterprises, device vendors, and others who require internally-scoped names to use sub-domains of registered public domain names whenever possible. However, this is not always feasible and there are legitimate use cases for private-use TLDs.

The need for private-use identifiers is not unique for domain names, and a useful analogy can be drawn between the uses of private IP address space and those of a private-use TLD. Network operators use private IP address space to number resources not intended to be externally accessible, and private-use TLDs are used by network operators in a similar fashion. This document proposes reserving a string in a manner similar to the current use of private IP address space. A similar rationale can be used to reserve more strings in case the need arises.

This document does not recommend a specific string for reservation. Instead, criteria are provided in Section 4.1 to guide the decision on which string to choose and assist the ICANN Board in making its determination. Four criteria are provided to help guide this decision and reasoning is provided for each.

This advisory takes a pragmatic approach to an issue that the DNS allows by its design. Because of the decentralized nature of the DNS, there is no way to prevent ad hoc use of a TLD, rather than use of an explicitly reserved private string as this advisory recommends. Nevertheless, the SSAC believes that the reservation of a private string will help to reduce the ad hoc usage, provide greater predictability for network administrators and equipment vendors, and, over time, reduce erroneous queries to root servers.

And after all of that bureaucratic boilerplate had settled down, ICANN wrote:

The Internet Assigned Numbers Authority (IANA) has made a provisional determination that ".INTERNAL" should be reserved for private-use and internal network applications. Prior to review and approval of this reservation by the ICANN Board, we are seeking feedback on whether the selection complies with the specified procedure from SAC113, and any other observations that this string would be an inappropriate selection for this purpose.

So it's all but certain that ".internal" will be reserved and will never be used for any public purpose and that it would be safe for anyone to start using it for any internal purpose.

The United Nations vs AI.

Last Thursday was a very busy day. The US Department of Justice announced that it plans to take down Apple, Apple's M-series processor silicon was called into question, and the United

Nations General Assembly adopted a resolution on artificial intelligence – not that anyone cares or that anyone can do anything about AI in any event. But for the record, UN officials formally called on tech companies to develop safe and reliable AI systems that comply with international human rights. And I loved this: "Systems that don't comply should be taken offline." Officials said the same rights that apply offline should also be protected online, including against AI systems.

I haven't said much about AI here since, just as I'm not trained as an attorney I do not have any expertise in AI systems. What I do have, however, is stunned amazement. As they would say over in the UK, I am gobsmacked by what I've seen. And what I may lack in expertise appears to have been made up for by my intuition, which has been screaming at me ever since I spent some time chatting with ChatGPT 4.

My take on the whole AI mess and controversy can be summed up in just four words, and they are: "good luck restraining anything." I doubt that any part of this is restrainable. At some point in the recent past we crossed over a tipping point and we're seeing something that no one would have believed possible even five years ago. Everyone knows there's no going back. Only people who have not been paying attention imagine that there's any hope of controlling what happens going forward. I don't know and I can't predict what the future holds, but whatever is going to happen, is going to happen... and I am pretty sure that it's bigger than us. We are not a sufficiently organized species to be able to contain or control this.

Telegram now blocked throughout Spain.

In a somewhat disturbing turn, Spain has joined the likes of China, Thailand, Pakistan, Iran, and Cuba to be blocking all use of and access to Telegram across its territory. This came after Spain's four largest media companies successfully complained that Telegram was being used to propagate their copyrighted content without permission. A judge with Spain's high court had asked Telegram to provide certain information relating to the case, and after Telegram chose not to respond to the judge's requests, he ordered all telecommunications carriers to block all access to Telegram throughout the country.

Vancouver Pwn2Own 2024

Last week, Vancouver held its 2024 Pwn2Own hacking competition. One security researcher by the name of Manfred Paul distinguished himself by successfully exploiting all four of the major web browsers, finding exploits in Chrome, Edge, Firefox, and Safari. He became this year's "Master of Pwn" and took home $202,500 dollars in prize money. Overall, the competing security researchers turned hackers successfully demonstrated 29 previously unknown 0-days during the contest and took home a total of $1.1 million in prize money. This also serves to demonstrate why I continue to believe that the best working model that's been presented for security is "porosity." We don't want it to be, but security is porous. How else can we explain that one lone research/hacker is able to take down all four of the industry's fully patched browsers whenever someone offers him some cash to do so? And that overall, 29 new previously unknown 0-days were revealed, when others were also offered some incentive.

You push hard and you can get in – that's the definition of porous.

I should also take a moment to give a shout out to Mozilla's Firefox team who had patched and updated Firefox fewer than 24 hours after the vulnerability was demonstrated. Frederik Braun posted on Mastodon: *"Last night, about 21 hours ago, Manfred Paul demonstrated a security exploit targeting Firefox 124 at pwn2own. In response, we have just published Firefox 124.0.1 (and Firefox ESR 115.9.1) containing the security fix. Please update your foxes! Kudos to all the countless people postponing their sleep and working towards resolving this so quickly! Really impressive teamwork again. Also, kudos to Manfred for pwning Firefox again :)"*

This is the way security is supposed to work. White Hat Hackers are given some reason to look, are compensated for their discoveries which make the products safer for everyone, and the publishers of those products promptly respond to provide all of that products' users the benefits of that discovery.

China warns of incoming hacks

Perhaps we (and others) are giving as good as we get. I've often noted that all we ever hear about are Chinese state-sponsored attacks against US critical infrastructure and that (naturally) we never hear about our similar successes against China. I want to believe that we would not be destructive if we could get in, and that we'd only seek to have a presence inside Chinese networks so that they understand that we're not sitting here defenseless. Well, it turns out that last week China's state security agency urged their local companies to improve their cybersecurity defenses. The Ministry of State Security said that foreign spy agencies have infiltrated hundreds of local businesses and government units. So that does sound like we may be at parity in this weird cyber cold war. I hate it, but it's what we've got.

Annual Tax Season Phishing Deluge

Just a reminder that here in the US it's tax season and that, as happens every year around this time, the level of targeted phishing scams increases. So everyone, just be a bit extra vigilant for the next several weeks.

SpinRite

I have a bunch of interesting news on the SpinRite front. One of the things that quickly became apparent as our listeners were wishing to obtain and use SpinRite v6.1 was that the world had changed in another way since SpinRite v6.0's release in 2004. Back then, Linux was still largely a curiosity with a relatively small fan base but no real adoption. Not so today. Back in 2004, it was acceptable to require a SpinRite user to use Windows to set up its boot media since Windows and Mac was all there was and SpinRite had never targeted the Mac market. Today, we've encountered many would-be users who do not have ready access to a Windows machine. So I needed to create a non-Windows setup facility that I have long envisioned but never needed until now. That now exists.

Over at GRC's /prerelease.htm page is the downloadable Windows/DOS executable and now also a downloadable ZIP file. The ZIP file, which is smaller than 400K, contains the image of the front of a 4GB FAT32 DOS partition. So any SpinRite owner without access to Windows may choose to instead download this ZIP file. It contains an 8.3 megabyte file named SR61.IMG which any

Linux user can "dd" copy onto any USB thumb drive to create an up to 4 gigabyte FAT32 partition that will immediately boot and run SpinRite. And the tricky bit that I worked out last week is that when this drive is booted for the first time, if the media onto which this image file was copied is smaller than the partition described by the image – for example, SpinRite's owner copied the image to an old but trusted 256 megabyte thumb drive – a little built-in utility named "downsize" will kick in, examine the size of the partition's underlying physical drive, and dynamically "downsize" the partition to fit onto its host drive. It's all transparent and automatic. And since this same technology will also be needed for SpinRite 7, it made sense to get it done.

The second new wrinkle to surface since last week is bad RAM. Over in GRC's web forums, a SpinRite 6.1 user reported data verification errors being produced by SpinRite when running on his cute little ZimaBoard. SpinRite always identified and logged the location of the apparent problem. But from one run to the next there was no correlation. And when he ran the same drive under SpinRite on a different PC, it passed SpinRite's most thorough Level 5 testing without a single complaint. And he was able to go back and forth to easily recreate this trouble multiple times on one system but not on the other.

The inhabitants of the forums jumped on this and suggested a bad or undersized power supply for his ZimaBoard, flaky cabling, and anything else that they could think of. All great suggestions. Finally, I asked this user to try running the venerable MemTest86 on his brand new ZimaBoard and guess what?! Yep. Memory errors. There should never be any. But the first time he ran MemTest86 it found 6 and the second time it found 101. Seeing that, we ran MemTest86 on our ZimaBoards and they all passed with zero errors.

So this user had a ZimaBoard with a marginal DRAM memory subsystem. There was no correlation in the locations of the errors from one memory test pass to the next, but there were two specific bits out of 32 that MemTest86 always identified as being the culprits. And SpinRite was getting tripped up by this machine's bad RAM when it was performing data verification that's available from SpinRite's levels 4 and 5. The problem was not the drive, it was the machine hosting SpinRite and the drive. So by this point our long-time listeners who have grown to know me probably know what I'm going to say next. That's right: SpinRite 6.1 now tests the memory of any machine it's running on.
SpinRite v1.0 also built-in a memory test. Back in 1988 it made sense to verify the RAM memory that would be used to temporarily hold a track's data while SpinRite was pattern testing the track and giving it a fresh new low-level format. But somewhere along the way I removed that feature from SpinRite. We never heard of it ever being useful, so my initially overly-cautious approach seemed to have been proven unnecessary... until last week.

So, late last week I implemented a very nice little DRAM memory tester for SpinRite and then had the guy with the bad ZimaBoard give it a try. It successfully determined that his machine's memory was not reliable and SpinRite will refuse to run on any such machine after that determination. It's just not safe to run it. And, of course, no such machine should be trusted for anything else. This new built-in RAM testing feature, which is not yet present in any SpinRite that's available to download, will appear along with a few other minor improvements shortly. I expect to announce it briefly next week.

Closing the Loop

Jazman / @jazmanua

Hi Steve, great show as always! I work in a cellphone free environment. Not only no service but we're not allowed to bring them. We have Internet computers, but we're not trusted to install anything on them. The problem is, I like to have 2FA to protect my email and other stuff. My understanding is that if I were to use passkeys I need my phone. I use Bitwarden with 2FA. My question, are there any good solutions for cell free environment? Kind regards Björn

In a phone-free environment I agree that relying upon Bitwarden for all authentication services is likely the best bet. As our listeners know. We would usually prefer to have passkeys or our authenticator on a separate device like a phone. But where that's not possible, merging those functions into a single password manager like Bitwarden makes sense. Note that Yubikeys are also passkeys capable and they can store up to 25 passkeys. So that's another possibility if the somewhat limited capacity doesn't pose a problem.

William Ruckman / @williamruckman

Hi Steve, Are Passkeys quantum safe? I thought public key crypto was vulnerable.

That's a terrific question because it's public key crypto such as that used by Passkeys that is expected to be vulnerable and since we're busy moving toward Passkeys while the world is moving toward post-Quantum cryptography. The answer is that the FIDO2 specification which underlies WebAuthn provides for plug-in future crypto. So Passkeys will be able to move to Quantum safe algorithms whenever that's appropriate.

GoFetch

Last Thursday the world learned that Apple had some problems with their cryptography. Unfortunately, it would be impossible to determine from most of the tech press' coverage of this whether this was an apocalyptic event or just another bump in the road. ArsTechnica was apparently unable to resist becoming click bait central with their headline *"Unpatchable vulnerability in Apple chip leaks secret encryption keys"* Wow! That would be bad if it was true. Fortunately it's not the least bit true. It's not unpatchable and it's not a vulnerability in an Apple chip. Kim Zetter's Zero Day goes with: *"Apple Chip Flaw Lets Hackers Steal Encryption Keys"* This "chip flaw" theme seems to have become pretty popular even though nowhere did any of the actual researchers ever say anything about any chip flaw. Even Apple Insider's headline read *"Apple Silicon vulnerability leaks encryption keys, and can't be patched easily"* What?! Apple was told 107 days before the disclosure, back on December 5th of last year. Apple is certainly quite aware of the issue and I'm sure they're taking it seriously. And for their newer M3 chips, all that's needed is for a single bit to be flipped. Tom's Hardware went with: *"New chip flaw hits Apple Silicon and steals cryptographic keys from system cache — 'GoFetch' vulnerability attacks Apple M1, M2, M3 processors, can't be fixed in hardware."* Oh dear! Except for a few details: It's not new. It's not a flaw. Nothing ever "hit" Apple Silicon. And as for it not being fixable in Apple M1, M2 or M3 processors, if you have an M3 chip, just flip the bit "on" during crypto operations and unfixable problem solved! And finally, as we'll see by the end of this topic today there are equally simple workarounds for the earlier M-series processors.

I'm tempted to keep going like this because the material in this instance is endless. Not a single one of the headlines of the supposedly tech press stories that covered this characterized this even close to accurately. It's not a flaw. Nothing is flawed. Everything is working just as it's supposed to. It's not a vulnerability in Apple silicon. Apple's silicon is just fine and nothing needs to change. And it's certainly not unfixable or unpatchable. CyberNews headline was *"M-series Macs can leak secrets due to inherent vulnerability"* The only thing that's inherently vulnerable here is the credibility of the tech press' coverage of this. It really has been quite over the top. After sitting back and thinking about this, the only explanation I can come up with is that because what's actually going on with this is wonderfully and subtly complex, no one writing for the press really understood what the researchers have very carefully and reasonably explained. So they just went with variations on ArsTechnica's "unpatchable vulnerability in Apple's chip" nonsense, under the assumption that Ars actually had understood what's going on so everyone just copied them. Fortunately, we have this podcast and I love nothing more than digging into the details to figure out – and then explain – exactly what is, and is not, going on.

The TL;DR of this whole fiasco is that a handful of researchers built upon an earlier 2-year-old discovery – which three of them had been participants in – that was dismissed at the time as being of only academic interest. It's yet another form of side-channel attack on otherwise very carefully designed (to be side-channel attack free) constant-time cryptographic algorithms. The attack surrounds an ARM-based performance optimization feature known as DMP. A variation of the same type of optimization is also present in newer Intel chips.

And so, true to Bruce Schneier's observation that attacks never get worse, they only ever get better, about a year and a half after that initial discovery which never amounted to much, it turned out that the presence of this DMP optimization feature actually did and does create an exploitable vulnerability that can be very cleverly leveraged to reveal a system's otherwise well-protected cryptographic secrets. After verifying that this was true, the researchers did the responsible thing by informing Apple and Apple fixed it. Unfortunately, that true story doesn't make for nearly as exciting a headline. So none of the hyperventilating tech press explained it this way.

One important thing that sets this apart from the similar and related Spectre and Meltdown vulnerabilities from yesteryear, is that this new exploitation of the DMP optimizer is not purely theoretical. All we had back in those early days of speculative execution vulnerabilities was a profound fear over what could be done. It was clear that Intel had never intended for their chips' internal operation to be probed in this fashion; and not much imagination was required to envision how this might be abused. But we lacked any concrete real world proof of concept. Not so today, and not even post-quantum crypto is safe from attack, since we're not attacking the strength of the crypto, the underlying keys are being directly leaked.

The GoFetch proof-of-concept app running on an Apple Mac connects to the targeted app which contains the secrets. It feeds the app a series of inputs that the app signs or decrypts, inducing it to perform cryptographic operations that require it to use its secrets. As it's doing this, the app monitors aspects of the processors' caches that it shares with the targeted app in order to obtain hints about the app's secret key that's being used to perform these cryptographic operations. So how bad is it?

As I mentioned, the attack works against both pre- and post-quantum encryption. The demo GoFetch app requires less than an hour to extract a 2048-bit RSA key and a little over two hours to extract a 2048-bit Diffie-Hellman key. The attack takes 54 minutes to extract the material required to assemble a Kyber-512 key and about 10 hours for a Dilithium-2 key, though some time is also required for offline processing of the raw data. In other words, it's an attack that's practical to employ in the real world.

Okay. So what exactly is DMP, what did the researchers discover, and how did they arrange to make their Macs give up the closely held secrets being hidden inside? The research paper is titled: *"**GoFetch:** Breaking Constant-Time Cryptographic Implementations Using Data Memory-Dependent Prefetchers"* That sounds more complex than it is. We have "Breaking Constant-Time Cryptographic Implementations" – we already know that a classic side-channel vulnerability which is often present in poorly written crypto implementations is for an algorithm to in any way change its behavior depending upon the secret key it's using. If that happens, the key-dependent behavior change can be used to infer some properties of the key. So the first portion of the title tells us that this attack is effective against properly written constant-time cryptographic implementations. So that's not where things got screwed up. The second part of the paper's title is "Using Data Memory-Dependent Prefetchers" – and that's what's new here.

If you guessed that the performance optimization technique known as DMP stands for "Data Memory-Dependent Prefetchers" you would be correct. Three of the seven co-authors of today's paper co-authored the earlier groundbreaking research two years ago which described their

reverse engineered discovery of this DMP facility residing inside Apple's M-series ARM- derived chips. Back then they raised and waved a flag around, noting that what this thing was doing seemed worrisome, but they stopped short of coming up with any way to actually extract information. And the information was made public. We don't know for sure that sophisticated intelligence agencies might not have picked upon this and turned it into a working exploit as has now happened. But we do know for sure that Apple apparently didn't give this much thought or concern since every one of their Mac with M-series chips was vulnerable to exploitation several years later.

I'm going to share today's research Abstract and Introduction since it's packed with information and valuable perspective. They wrote:

> *Microarchitectural side-channel attacks have shaken the foundations of modern processor design. The cornerstone defense against these attacks has been to ensure that security-critical programs do not use secret-dependent data as addresses. Put simply: do not pass secrets as addresses to, e.g., data memory instructions. Yet, the discovery of data memory-dependent prefetchers (DMPs)—which turn program data into addresses directly from within the memory system—calls into question whether this approach will continue to remain secure.*
>
> *This paper shows that the security threat from DMPs is significantly worse than previously thought and demonstrates the first end-to-end attacks on security-critical software using the Apple m-series DMP. Undergirding our attacks is a new understanding of how DMPs behave which shows, among other things, that the Apple DMP will activate on behalf of any victim program and attempt to "leak" any cached data that resembles a pointer. From this understanding, we design a new type of chosen-input attack that uses the DMP to perform end-to-end key extraction on popular constant-time implementations of classical and post-quantum cryptography.*

And here's their introduction:

> *For over a decade, modern processors have faced a myriad of microarchitectural side-channel attacks, e.g., through the caches, TLBs, branch predictors, on-chip interconnects, memory management units, speculative execution, voltage/frequency scaling and more.*
>
> *The most prominent class of these attacks occurs when the program's memory access pattern becomes dependent on secret data. For example, cache and TLB side-channel attacks arise when the program's data memory access pattern becomes secret dependent. Other attacks, e.g., those monitoring on-chip interconnects, can be viewed similarly with respect to the program's instruction memory access pattern. This has led to the development of a wide range of defenses—including the ubiquitous constant-time programming model, information flow-based tracking, and more — all of which seek to prevent secret data from being used as an address to memory/control-flow instructions.*
>
> *Recently, however, Augury [that was the earlier research two years ago] demonstrated that Apple M-series CPUs undermine this programming model by introducing a Data Memory-Dependent Prefetcher (DMP) that will attempt to prefetch addresses found in the contents of program memory. Thus, in theory, Apple's DMP leaks memory contents via cache side channels, even if that memory is never passed as an address to a memory/control-flow instruction.*

[I'll explain exactly what all that means in a minute.]

Despite the Apple DMP's novel leakage capabilities, its restrictive behavior has prevented it from being used in attacks. In particular, Augury reported that the DMP only activates in the presence of a rather idiosyncratic program memory access pattern (where the program streams through an array of pointers and architecturally dereferences those pointers). This access pattern is not typically found in security critical software such as side-channel hardened constant-time code — hence making that code impervious to leakage through the DMP.

With the DMP's full security implications unclear, in this paper we address the following [two] questions: Do DMPs create a critical security threat to high-value software? Can attacks use DMPs to bypass side-channel countermeasures such as constant-time programming?

This paper answers the above questions in the affirmative, showing how Apple's DMP implementation poses severe risks to the constant-time coding paradigm. In particular, we demonstrate end-to-end key extraction attacks against four state-of-the-art cryptographic implementations, all deploying constant-time programming.

As we've had the occasion to discuss through the years on this podcast, the performance of DRAM, the dynamic RAM memory that forms the bulk of our systems' memory has lagged far behind the memory bandwidth demands of our processors. Through the years we've been able to significantly increase the density of DRAM, but not its performance. And, as we know, even the increase in density was met with challenges in the form of susceptibility to adjacent row interference which led to the various DRAM hammering attacks.

But on the performance side, the saving grace has been that processor memory access patterns are not linear and non-repetitive – they are typically highly repetitive. Programs almost always "loop", meaning that they are executing the same code again and again, over and over. And that, in turn, means that if a much smaller but much faster "cache" of memory is inserted between the main DRAM and the processor, the processor's repetition of the same instructions and the data for those instructions, can be fulfilled much more quickly from the cache than from main memory.

During our discussions of speculative execution we saw that another way to speed up our processors was to allow the processor to run well ahead of where execution was and if the code encountered a fork in the road it would fetch ahead down both paths of the fork so that once the path to be taken became know, whichever way that went, the system would already have read the coded instructions for that path and have them ready. In practice this is accomplished by breaking our processors into several specialized pieces, one being the pre-fetch engine whose job it is to keep the execution engines fed with data from main memory.

Many instructions do not make any main memory accesses. They might be working only within the processor's internal registers or with what's already present in the processor's local cache. So this gives the pre-fetch engine time to anticipate where the processor might go next and what it might need. In a modern system, there's never any reason to allow main memory to sit idly by. Not even for a single cycle. A good pre-fetching system will always be working to anticipate its processor's needs and to have already loaded the contents of slower DRAM into the

high speed cache.

Now to this let's add one additional layer of complexity: One of the features of all modern processor architectures is the concept of a pointer. A location in memory or the contents of a register could contain an object's value itself, or instead it could contain the memory address of the object. In that second case we would say that the value in the memory or register contains, instead of the value of the object itself, a pointer to the object. As a coder, I cannot imagine life without pointers. They are absolutely everywhere in code and they are so useful.

We need one bit of new vocabulary to talk about pointers. Since a pointer is used to point to or refer to something else, the pointer contains a reference to the object. So the act of following a pointer to the object is known as "dereferencing" the pointer. We'll see the researchers using that jargon in a minute.

But first, let's think about that cache-filling pre-fetch engine. Its entire reason for existence is to anticipate the future needs of its processor so that whatever the processor wants will already be waiting for it and instantly available from its cache. The processor will think that its pre-fetch engine is magic.

So one evening, probably about seven years ago, some Apple engineers are sitting around a whiteboard with a bunch of half eaten pizzas. They're brainstorming ways to further speed up Apple's proprietary silicon. Given the time frame, this would first be able to appear in their A14 Bionic processor. And one of them says: "You know, we're already doing a great job of fetching the data that the processor is going to ask for. But when we fetch data that contains what looks like pointers, we're not fetching the data that those pointers are pointing to. If the data really are pointers then there's a good chance that once the processor gets its hands on them, it's going to be asking for that data next. We could anticipate that and have it ready, too, just in case it might be useful. I mean... that's the whole point of being a pre-fetching engine, right?"

At this point, the pizza is forgotten and several in the group lean forward. They're thinking about... ... ... the kinds of cars they're going to be able to get with the raises this idea will earn them. Then they realize they'll need to make it work, first.

Although they're immediately hooked by the idea because they know there's something there, one of them plays devil's advocate, saying: "But the cache is context-free." What he means by that is that to the pre-fetch engine everything is just data. It's all the same. The pre-fetcher doesn't know what the data means – it has no meaning in DRAM. It's all just mixed bytes of instructions and data. It's a hodgepodge. It's not until that data is fetched from the cache and is actually consumed by the processor that the data acquires context and meaning.

The answer to the "cache is context-free" guy is "yeah... and so what?" If some data that's being added to the cache looks like a pointer, and if it's pointing into valid DRAM memory, what's the harm in treating it **as** a pointer and going out and also grabbing the thing that it **might** be pointing to? If we have time and we're right, it's a win for the processor. The processor won't believe its luck in already having the thing it was just about to ask for already, magically, waiting there in its local cache.

So, finally, after their last dry-erase marker stops working from the hastily scribbled diagrams on their whiteboards, they're satisfied that they're really onto a useful next-generation optimization. So one of them asks: "Okay. This is good. But it needs a name. What are we going to call it?" Uhhhhh... how about a "Data Memory-Dependent Prefetcher" or DMP for short?

So here we've just seen a perfect example of where and how these next-generation features are invented – over pizza and dry-erase markers. And it's also easy to see that the security implications of this aren't even on the radar. All they're doing, after all, is anticipating a possible future use of what might be a pointer and pre-fetching the thing it's pointing to in case they're right and it is a pointer and in case the processor might eventually ask for it. It's disconnected from whatever the processor is doing, right? It's a Data Memory-Dependent Prefetcher. What this amounts to is a somewhat smarter prefetcher. It cannot be certain whether it's fetching a pointer. But in case it might be, it'll jump ahead even further to also prefetch the thing that what might be a pointer may be pointing to.

Okay. So now let's hear from the geniuses who likely also consumed their share of pizza while they scratched the itch that had apparently been lingering with at least three of them for a couple of years, ever since they first discovered that Apple had dropped this Data Memory-Dependent Prefetcher into their silicon. Here's how they explain what they came up with:

> *We start by reexamining the findings in Augury. Here we find that Augury's analysis of the DMP activation model was overly restrictive and missed several DMP activation scenarios. Through new reverse engineering, we find that the DMP activates on behalf of potentially any program, and attempts to dereference **any** data brought into cache that resembles a pointer.*
>
> *This behavior places a significant amount of program data at risk, and eliminates the restrictions reported by prior work. Finally, going beyond Apple we confirm the existence of a similar DMP on Intel's latest 13th generation (Raptor Lake) architecture with more restrictive activation criteria.*
>
> *Next, we show how to exploit the DMP to break security-critical software. We demonstrate the widespread presence of code vulnerable to DMP-aided attacks in state-of-the-art constant-time cryptographic software, spanning classical to post-quantum key exchange and signing algorithms.*

And here's the key to everything. I'll read it first then take it apart:

> *Our key insight is that while the DMP only dereferences pointers, an attacker can craft program inputs so that when those inputs mix with cryptographic secrets, the resulting intermediate state can be engineered to **look like a pointer** if and only if the secret satisfies an attacker-chosen predicate. For example, imagine that a program has secret s, takes x as input and computes and then stores $y = s \oplus x$ to its program memory. The attacker can craft different x and infer partial (or even complete) information about s by observing whether the DMP is able to dereference y. We first use this observation to break the guarantees of a standard constant-time swap primitive recommended for use in cryptographic implementations. We then show how to break complete cryptographic implementations designed to be secure against chosen-input attacks.*

Okay. So they realized that Apple's DMP technology is far more aggressive than they initially appreciated. It is busily examining all of the data that's being put into the cache for all of the processes running in the system. It's looking for anything "pointer-like" and, when found, it's going out to pre-fetch that "maybe pointed to" data, too.

Their next step was to realize that since this "pointer-like" behavior is highly prone to producing false positive hits which would prefetch miscellaneous bogus data, and since it operates indiscriminately on any and all data in the system, they can deliberately trick Apple's DMP system to misfire. When it does, it will prefetch data that wasn't really being pointed to and they can use standard well-understood cache probing to determine whether or not the DMP did misfire and prefetch.

Finally, they induce the isolated process containing the secrets to perform a large number of cryptographic operations on their deliberately crafted data while using the now-well- understood behavior of the DMP to create an inadvertent side channel that leaks the secret key even though the cryptographic code is being super-careful not to behave differently in any way based upon the value of the secret key. That doesn't matter because the cryptographic code is being betrayed by this overly helpful cache prefetching system.

What I've just explained is a version of what these very clever researchers revealed to Apple back in early December last year. So what does Apple do about this? This does seem like the sort of thing that Apple ought to be able to turn off. One of the things we've learned is that these initially nifty seeming slick performance optimizations always seem to come back to bite us sooner or later. So anything like this should really have an "off" switch.

And, what do you know? It may have been (and likely was) in reaction to these researcher's initial Augury DMP paper back in 2022, but Apple's M3 chip, which Apple announced the day before Halloween on October 30th last year, contains exactly that off switch. I've heard, but haven't confirmed, that Apple's own crypto code is flipping DMP off during any and all of their own cryptographic operations. So it may only be non-Apple crypto code running on Macs that are endangered on M3-based Macs. The researchers cite their compromise of the Diffie-Hellman Key Exchange in OpenSSL, and the RSA key in the Go language library for RSA crypto.

But what about the non-M3 chips? The Apple A14 Bionic, the M1 and M2?

Well, it turns out that these so-called SoC or Systems on a Chip all have multiple cores and the cores are not all of the same type – only half of the cores are vulnerable. Apple's M-series have two types of cores: the bigger Firestorm cores also known as the performance cores and the smaller Icestorm cores, also known as the efficiency cores. On the M1 and M2 chips only the Firestorm performance cores offer the problematic DMP prefetching system. So all Apple needs to do is to move their crypto over to the smaller efficiency cores. Crypto operations will run more slowly there, but they will be completely secure from this trouble.

So... is Apple going to do any of these things? Have they already? The press thinks that nothing has been done yet. I find that curious given that the concerns are real and that solutions are available. But so far, all of the press has reported that Apple has been curiously mute on this subject. They just say "no comment." This is doubly confounding given that Thursday's research

disclosure came as no surprise to them and that the firestorm of truly over the top apoplectic and apocalyptic headlines that have ensued really need some response. I imagine that something will be forthcoming from Apple, soon.

Until then, the attack, if it were to happen, would be local and targeted and would require someone arranging to install software on the victim's machine. It's not the end of the world, and as I'm always saying around here, anyone can make a mistake. But Apple's customers would seem to need and deserve more than silence from Apple.