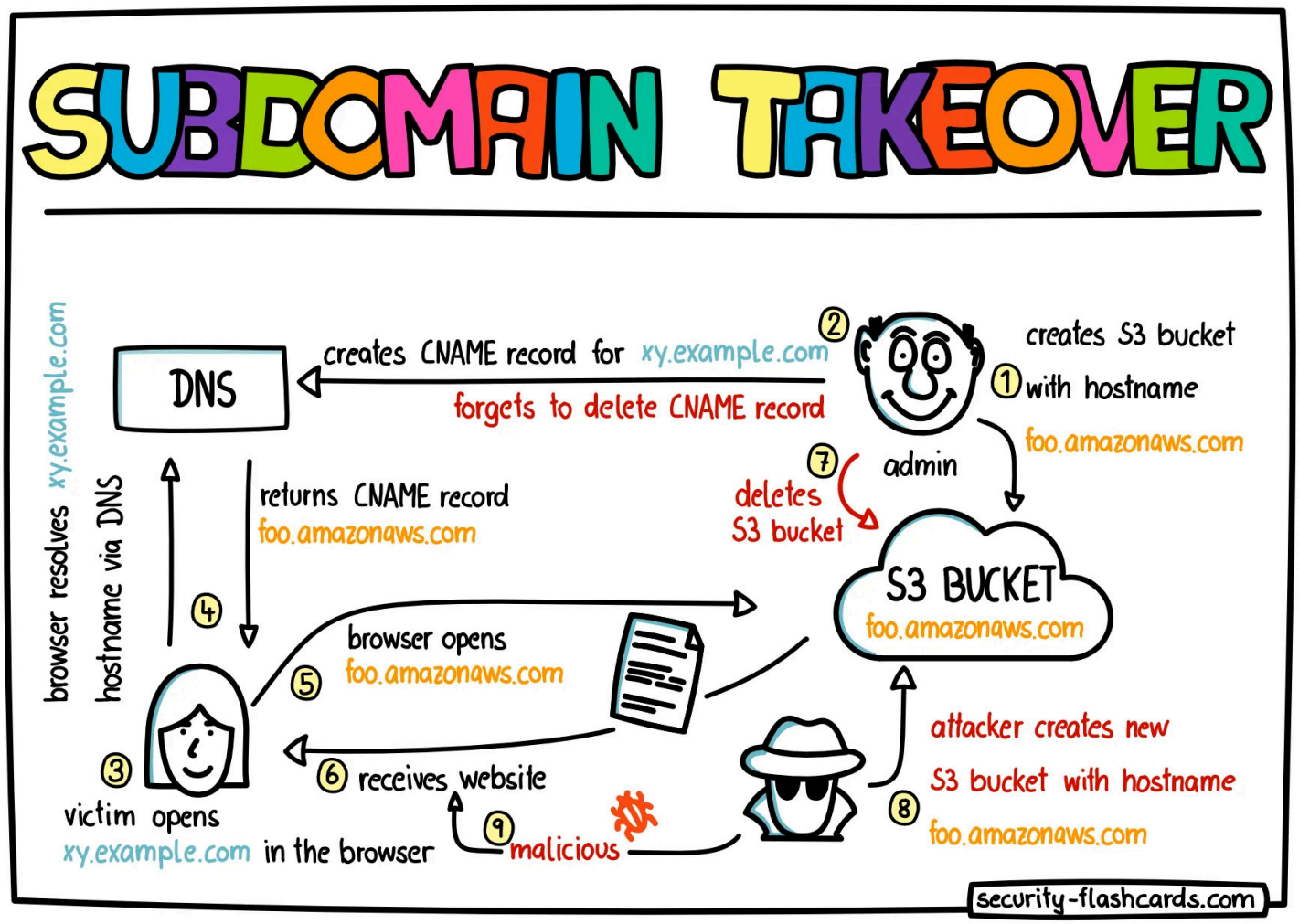
Security Now! #979 - 06-18-24

The Angle of the Dangle

This week on Security Now!

Why is updating your Windows laptop with last week's patches potentially much more important than usual? CoPilot+'s Recall feature won't be released today; what happened? Was Recall recalled? What does Johns Hopkins well-known cryptographer think about Apple's new Private Cloud Compute concept? How could the WGET command-line utility possibly have a CVSS 10.0 vulnerability? Or does it? What order did Google, Cloudflare and Cisco recently receive from a Parisian court? And after a brief GRC email update and three pieces of closing the loop feedback from our listeners, we're going to examine exactly how Microsoft lost control of their code.microsoft.com subdomain and why the underlying problem is far bigger than them.

Fortunately, as we'll see today, the "Subdomain Takeover" problem is much less confusing than this diagram!



Security News

CVE-2024-30078

I need to begin this week by making 100% certain that everyone listening is aware of a flaw that was patched during last Tuesday's Windows patchfest. It's CVE-2024-30078 and the only reason it only has a CVSS score of 8.8, rather than 11 on a scale that maxes out at 10, is that an attacker needs to be within WiFi radio range of the Windows machine – any Windows machine with WiFi enabled that has not been updated with last Tuesday's patches – in order to obtain total control over that machine. Yes, that's right. Last Tuesday eliminated a fortunately very rare remote wireless radio takeover of **any** Windows machine which is using its native WiFi protocol stack. This affects all versions of Windows ever, so that means that any machine that was **not** updated for whatever reason last Tuesday, remains vulnerable today. Microsoft is only saying "all supported versions of Windows", but that typically means all earlier versions, too. And the fact that this appears to be Windows-wide suggests that the flaw was in a core component that they haven't been messing with.

To provide some additional color, background and perspective I'll share what Forbes cybersecurity guy wrote under their headline "New Wi-Fi Takeover Attack—All Windows Users Warned To Update Now":

Microsoft has confirmed a new and quite alarming Wi-Fi vulnerability in Windows, which has been rated 8.8 out of 10 in terms of severity using the Common Vulnerability Scoring System. The vulnerability, assigned as CVE-2024-30078, does not require an attacker to have physical access to the targeted computer, although physical proximity is needed. Exploiting this vulnerability can allow an unauthenticated attacker to gain remote code execution on the impacted device. What's perhaps most concerning, though, is that this Wi-Fi driver security flaw affects all supported versions of the Windows operating system.

Microsoft has confirmed that with no special access conditions or extenuating circumstances are needed, apart from the proximity requirement, an attacker could "expect repeatable success against the vulnerable component." Microsoft also warns that an attacker requires no authentication as a user before exploiting this vulnerability, nor any access to settings or files on the victim's machine before carrying out the attack. Furthermore, the user of the targeted device does not need to interact at all: there is no link to click, no image to load, and no file to execute.

Jason Kikta, chief information security officer at Automox, said that, given its nature, "this vulnerability poses a significant risk in endpoint-dense environments including hotels, trade shows, or anywhere else numerous devices connect to WiFi networks." In these kinds of environments, it would be all too easy for an attacker to target users without raising any red flags. To protect against this vulnerability it's recommended that you apply the latest patches as soon as possible.

Assuming, that is, you are using a version of Windows that still receives security updates. Anyone using an end-of-life version of Windows without an extended service contract is recommended to update to a supported version as soon as possible Kikta said: "If patching immediately isn't feasible, you must use endpoint detection to monitor for suspicious activity related to this vulnerability. Due to its unique nature, it is unlikely to be visible to network-level detection methods." He added: "The risk of running outdated software cannot be overstated."

The article concludes:

> *In case you need any further incentive to get patching as soon as possible, Kikta said: "This close access vector threat potentially bypasses network-based detections and mitigations. It circumvents most threat modeling, so this is an immediate patch priority for me." Most security experts agree that publicly available exploitation tools will be available before long, so the window of opportunity to patch before the attacks start is getting smaller all the time.*

And on that note, Github does indeed already have a vulnerability detection and command execution script posted. Microsoft's own page for tracking this CVE enumerates the following characteristics of this vulnerability:

> ***Attack Vector: Adjacent***
> *The vulnerable component is bound to the network stack, but the attack is limited at the protocol level to a logically adjacent topology. This can mean an attack must be launched from the same shared physical (e.g., Bluetooth or IEEE 802.11) or logical (e.g., local IP subnet) network, or from within a secure or otherwise limited administrative domain.*
>
> ***Attack Complexity: Low***
> *Specialized access conditions or extenuating circumstances do not exist. An attacker can expect repeatable success against the vulnerable component.*
>
> ***Privileges Required: None***
> *The attacker is unauthorized prior to attack, and therefore does not require any access to settings or files to carry out an attack.*
>
> ***User Interaction: None***
> *The vulnerable system can be exploited without any interaction from any user.*
>
> ***Confidentiality: High Threat***
> *There is total loss of confidentiality, resulting in all resources within the impacted component being divulged to the attacker.*
>
> ***Integrity: High Threat***
> *There is a total loss of integrity, or a complete loss of protection. For example, the attacker is able to modify any/all files protected by the impacted component.*

Microsoft's tracking of this CVE includes two FAQ questions and answers:

> ***Q:*** *According to the CVSS metric, the attack vector is adjacent (AV:A). What does that mean for this vulnerability?* ***A:*** *Exploiting this vulnerability requires an attacker to be within proximity of the target system to send and receive radio transmissions.*
>
> ***Q:*** *How could an attacker exploit the vulnerability?* ***A:*** *An unauthenticated attacker could send a malicious networking packet to an adjacent system that is employing a Wi-Fi networking adapter, which could enable remote code execution.*

The appearance of this vulnerability provides a perfect case-in-point demonstration of why the presence of Recall running in Windows machines represents the threat that it does. I have no doubt that Microsoft's heart is in the right place; they are not an evil empire. But they're now attempting to add an extremely powerful and high-risk feature to an old, creaky and bug-ridden operating system. This vulnerability demonstrates what we all intuitively feel, which is that Windows is not up to the task of protecting anything so valuable.

This vulnerability was reported by a prolific Chinese security researcher who has helped Microsoft in the past. What we don't know is what, if any, other agencies may have known about this beforehand and chose to keep quiet because it might come in handy for their own use.

My favorite Tweeted comment from someone who had just learned that this WiFi vulnerability allowed for a complete zero-click remote wireless Windows takeover was *"But don't worry, your Recall data is safe!"* Which brings us to some additional welcome news on the Recall front:

"Recall" has been recalled

First they switched "Recall" to Opt-In – that was a welcome move. But as we know, I was still quite worried that they would nevertheless be promoting the heck out of it, and why would Microsoft ever mention the very real danger inherent in having any machine storing its entire history? So the latest good news is that Recall has now been completely removed from today's June 18th production release of Windows. It will initially only be available to users participating in the Windows Insider Program.

Last week, I shared Microsoft's blog posting where, amid all of their mumbo jumbo about how they're putting security first, they explained that they would be switching Recall to Opt-In. In an update to that blog posting last Thursday, they wrote:

> *Update: June 13, 2024: Today, we are communicating an additional update on the Recall (preview) feature for Copilot+ PCs. Recall will now shift from a preview experience broadly available for Copilot+ PCs on June 18, 2024, to a preview available first in the Windows Insider Program (WIP) in the coming weeks. Following receiving feedback on Recall from our Windows Insider Community, as we typically do, we plan to make Recall (preview) available for all Copilot+ PCs coming soon.*
>
> *We are adjusting the release model for Recall to leverage the expertise of the Windows Insider community to ensure the experience meets our high standards for quality and security. This decision is rooted in our commitment to providing a trusted, secure and robust experience for all customers and to seek additional feedback prior to making the feature available to all Copilot+ PC users.*

So this is terrific news. Microsoft will get there eventually if it wishes to. And I'm quite certain that it wishes to. As I said, I believe Microsoft has more in store for a machine's entire usage history than just scrolling back through time. Since the actual delivered security of anything can only be judged retrospectively, that means that only time will tell whether they have been able to sufficiently protect any machine's entire usage history. What they were on the brink of launching was obviously only half-baked, if that. So it appears that they allowed their enthusiasm for this new capability to get the better of them. And as we saw from the reactions

of the entire industry, this sort of immature exuberance does not inspire confidence. The main point I've wanted to make is to note that we've never seen anything like Recall before. It is not just a change in degree – this represents a change in kind – and it presents a security challenge of an entirely different scale.

And speaking of security challenges, this brings us to Matthew Green's multi-part Mastodon posting thread in reaction to Apple's announcement of their "Private Cloud Compute" system:

Matthew Green on Apple's Private Cloud Compute
(To remind of Matthew's pedigree, he's an American cryptographer and security technologist, Associate Professor of Computer Science at the Johns Hopkins Information Security Institute.)

So Apple has introduced a new system called "Private Cloud Compute" that allows your phone to offload complex (typically AI) tasks to specialized secure devices in the cloud. I'm still trying to work out what I think about this. So here's a thread.

Apple, unlike most other mobile providers, has traditionally done a lot of processing on-device. For example, all of the machine learning and OCR text recognition on Photos is done right on your device.

The problem is that while modern phone "neural" hardware is improving, it's not improving fast enough to take advantage of all the crazy features Silicon Valley wants from modern AI, including generative AI and its ilk. This fundamentally requires servers.

But if you send your tasks out to servers in "the cloud" this means sending incredibly private data off your phone and out over the Internet. That exposes you to spying, hacking, and the data hungry business model of Silicon Valley.

The solution Apple has come up with is to try to build secure and trustworthy hardware in their own data centers. Your phone can then "outsource" heavy tasks to this hardware. Seems easy, right? Well: here's the blog post.

https://security.apple.com/blog/private-cloud-compute/

***TL;DR: it is not easy.** Building trustworthy computers is literally the hardest problem in computer security. Honestly, it's almost the **only** problem in computer security. But while it remains a challenging problem, we've made a lot of advances. Apple is using almost all of them.*

The first thing Apple is doing is using all the advances they've made in building secure phones and PCs in their new servers. This involves using Secure Boot and a Secure Enclave Processor (SEP) to hold keys. They've presumably turned on all the processor security features.

Then they're throwing all kinds of processes at the server hardware to make sure the hardware isn't tampered with. I can't tell if this prevents hardware attacks, but it seems like a start.

Matthew then includes a screen shot from Apple's posting, which explains:

Private Cloud Compute hardware security starts at manufacturing, where we inventory and perform high-resolution imaging of the components of the Private Cloud Compute node before each server is sealed and its tamper switch is activated. When they arrive in the data center, we perform extensive revalidation before the servers are allowed to be provisioned for PCC. The process involves multiple Apple teams that cross-check data from independent sources, and the process is further monitored by a third-party observer not affiliated with Apple. At the end, a certificate is issued for keys rooted in the Secure Enclave UID for each PCC node. The user's device will not send data to any PCC nodes if it cannot validate their certificates.

Okay, so I need to confess that I'm a bit of a fanboy for the idea that Apple is performing high-resolution imaging of each node's components in order to detect anything that might have been done to the server between design and through manufacturing. That's very cool. And just the fact that it's now widely known that this is being done, likely serves to prevent anyone from even trying to do it. Matt continued...

They also use a bunch of protections to ensure that software is legitimate. One is that the software is "stateless" and allegedly doesn't retain any information between user requests. To help ensure this, each server/node reboot re-keys and wipes all storage.

Again, his quotes Apple's announcement:

We designed Private Cloud Compute to make several guarantees about the way it handles user data:

- *A user's device sends data to PCC for the sole, exclusive purpose of fulfilling the user's inference request. PCC uses that data only to perform the operations requested by the user.*
- *User data stays on the PCC nodes that are processing the request only until the response is returned. PCC deletes the user's data after fulfilling the request, and no user data is retained in any form after the response is returned.*
- *User data is never available to Apple — even to staff with administrative access to the production service or hardware.*

Matt continues:

A second protection is that the operating system can "attest" to the software image it's running. Specifically, it signs a hash of the software and shares this with every phone/client. If you trust this infrastructure, you'll know it's running a specific piece of software.

Of course, knowing that the phone is running a specific piece of software doesn't help you if you don't trust the software. So Apple plans to put each binary image into a "transparency log" and publish the software. But here's a sticky point: not with the full source code.

Matt quote Apple:

Our commitment to verifiable transparency includes:

1. *Publishing the measurements of all code running on PCC in an append-only and cryptographically tamper-proof transparency log.*

2. *Making the log and associated binary software images publicly available for inspection and validation by privacy and security experts.*

3. *Publishing and maintaining an official set of tools for researchers analyzing PCC node software.*

4. *Rewarding important research findings through the Apple Security Bounty program.*

Matt says:

*Security researchers will get *some code* and a VM they can use to run the software. They'll then have to reverse-engineer the binaries to see if they're doing unexpected things. It's a little suboptimal.*

When your phone wants to outsource a task, it will contact Apple and obtain a list of servers/nodes and their keys. It will then encrypt its request to all servers, and one will process it. They're even using fancy anonymous credentials and a third party relay to hide your IP [from themselves.]

Quoting Apple about this:

Target diffusion starts with the request metadata, which leaves out any personally identifiable information about the source device or user, and includes only limited contextual data about the request that's required to enable routing to the appropriate model. This metadata is the only part of the user's request that is available to load balancers and other data center components running outside of the PCC trust boundary. The metadata also includes a single-use credential, based on RSA Blind Signatures, to authorize valid requests without tying them to a specific user. Additionally, PCC requests go through an OHTTP relay — operated by a third party — which hides the device's source IP address before the request ever reaches the PCC infrastructure. This prevents an attacker from using an IP address to identify requests or associate them with an individual. It also means that an attacker would have to compromise both the third-party relay and our load balancer to steer traffic based on the source IP address.

Matt says:

Ok there are probably half a dozen more technical details in the blog post. It's a very thoughtful design. Indeed, if you gave an excellent team a huge pile of money and told them to build the best "private" cloud in the world, it would probably look like this.

***But now the tough questions.** Is it a good idea? And is it as secure as what Apple does today? And most importantly: Can users opt-out entirely from this feature?*

I admit that as I learned about this feature, it made me kind of sad. The thought that was going through my head was: this is going to be too much of a temptation? Once you can "safely" outsource tasks to the cloud, why bother doing them locally. Outsource everything!

As best I can tell, Apple does not have explicit plans to announce when your data is going off-device to Private Compute. You won't opt into this, you won't necessarily even be told it's happening. It will just happen. Magically. I don't love that part.

Finally, there are so many invisible sharp edges that could exist in a system like this. Hardware flaws. Issues with the cryptographic attestation framework. Clever software exploits. Many of these will be hard for security researchers to detect. That worries me too.

Wrapping up on a more positive note: it's worth keeping in mind that sometimes the perfect is the enemy of the really good. In practice the alternative to on-device is: ship private data to OpenAI or someplace sketchier, where who knows what might happen to it.

And of course, keep in mind that super-spies are not your biggest adversary. For many people your biggest adversary is the company who sold you your device/software. This PCC system represents a real commitment by Apple not to "peek" at your data. That's a big deal.

In any case, this is the world we're moving to. Your phone might seem to be in your pocket, but a part of it lives 2,000 miles away in a data center. As security folks, we probably need to get used to that fact, and do the best we can to make sure all parts are secure.

I think Matthew's take is exactly right. The design of this system is what you would get if a bunch of very good engineers and cryptologists were to deliberately design a system that was meant to transiently extend an individual smartphone's local computing into the cloud for the purpose of performing some very heavy lifting. It takes advantage of everything we know about how to do this safely and securely. It will enable Apple's devices to do things no other mobile devices can do.

I have a concern that Matt did not raise, which is that because Apple has made this transparent to their users, no one will be able to appreciate the lengths Apple has gone to, to securely offer this capability. The listeners of this podcast understand that Apple is visually inspecting the motherboards of their servers prior to deployment because, for example, we've covered the worries over tiny chips being added to server hardware or Cisco routers when they're intercepted during transhipping. Even though that's way out there, it's a factor Apple has preemptively considered. Who else is going to go to these extremes?

It's not that I'm worried about Apple being underappreciated. It's that I can easily see "me too" vendors popping up and offering their own outwardly similar capabilities that APPEAR to be the same, while providing none of the same true protections. They'll be able to say: *"We're doing the same thing Apple is doing"*, thus riding on Apple's coattails while providing far less true security for their users, at a far lower cost to themselves.

The concern is that Apple is legitimizing and popularizing the idea of exporting what could be extremely personal mobile device data to the cloud for cloud-based processing. Other vendors are going to do the same. But users of those look alike services will be placing their users' data at far greater risk.

A WGET flaw with a CVSS of 10.0?
There's buzz in the industry today of a recently discovered flaw in the widely used WGET command line utility. Some outlets are claiming that this flaw carries an attention-getting CVSS score of 10.0, but anyone reading that should be immediately skeptical. As we've seen, 10.0 scores are pretty much reserved for "end of the world as we've known it" flaws, and it's hard to see how you can have an end of the world flaw that's not remotely exploitable, and probably also wormable without any user interaction at the receiving end. But WGET is just a convenient command-line tool used to retrieve files. I use it every week to grab this podcast's audio for recompression, before I post it to Elaine for transcription. So how any flaw in any command-line tool that's not publicly exposing a vulnerable service on the Internet is beyond me.

So I did some digging and it's true that there's a problem with WGET up through version 1.24.5. The problem surrounds incorrect parsing of semicolons appearing in the "userinfo" portion of the URL that's passed to WGET. Recall that URLs are able to carry a username and password which appears before the hostname. So rather than https://example.com, you could have https://username:password@example.com. Therefore, the concern is that mishandled semicolons might lead WGET's parser to confuse the userinfo with the hostname in some way. I located the dialog with the guy who patched this a few weeks ago. He noted:

> *I just pushed a fix for the issue. Indeed, the URL parser implementation of wget 1.x is based on RFC 2396, a standard from 1998. But even by this outdated standard, the implementation of the userinfo parsing was not correct. It hopefully is correct now.*
>
> *Anyway, nobody is going to lift the whole URL parsing of wget 1.x to newer standards. But we have wget2, and Fedora 40 recently switched to using wget2 instead of wget (of course there are corner cases that break backward compatibility). Regards, Tim*

So if you see anyone running around screaming about a CVSS of 10.0 in WGET while looking up to see whether the sky is falling, you can put their mind at ease. All anyone ever had was a concern raised by seeing that semicolons were being mishandled. The CVE for this minor parsing flaw appears to have just been assigned and published last Saturday, June 15th, so it's quite recent. NIST's National Vulnerability Database lists the CVE but doesn't yet have any CVSS assigned yet. As I was going over this again just before the podcast I found a new reference at Red Hat which lists this with a CVSS of 5.4, which is far more sane.
https://access.redhat.com/security/cve/cve-2024-38428

Thou shall not Resolve!
As a result of a lawsuit recently brought by Canal+, a French sports broadcaster, a French court has ordered three popular public DNS providers Google, Cloudflare and Cisco to selectively edit their DNS domain name resolution in order to block the lookup of around 117 individual pirate sports streaming domain names. Why not just sue copper wire for having the audacity to carry electrons? We've covered this sort of conduct before, and it's just as wrong now as it was then.

TorrentFreak posted an article last Thursday which explained how this battle has been slowly escalating for the past year. They wrote:

In 2023, Canal+ went to court in France to tackle pirate sports streaming sites including Footybite.co, Streamcheck.link, SportBay.sx, TVFutbol.info, and Catchystream.com. Canal+ said that said that since subscribers of local ISPs were accessing the pirate sites using their Internet services, the ISPs should prevent them from doing so.

When the court agreed, all major French ISPs were required to implement technical measures to comply. Since the ISPs have their own DNS resolvers for use by their own customers, these were configured to provide non-authentic responses to deny access to the sites in question.

Naturally, in response to this blackout, increasingly savvy internet users that hadn't already done so simply changed their settings to use different DNS providers – Cloudflare, Google, and Cisco – whose resolvers hadn't been tampered with; at least not yet.

Use of third-party DNS providers to circumvent blocking is common. So last year Canal+ took legal action against three popular public DNS providers – Cloudflare (1.1.1.1), Google (8.8.8.8), and Cisco (208.69.38.205), in each case demanding measures similar to those which had already been implemented by French ISPs. And once again the court agreed.

[TorrentFreak writes that:] Tampering with public DNS is a step too far for many internet advocates. But for major rightsholders, if the law can be shaped to allow it, that's what will happen. In this case, Article L333-10 of the French Sports Code (which became active in January of 2022) seems capable of accommodating almost anything.

It reads, when there are "serious and repeated violations" by an "online public communication service" whose main objective is the unauthorized broadcasting of sports competitions, rightsholders can demand "all proportionate measures likely to prevent or put an end to this infringement, against any person likely to contribute to remedying it."

Two decisions were handed down by the Paris judicial court last month; one concerning Premier League matches and the other the Champions League. The orders instruct Google, Cloudflare, and Cisco to implement measures similar to those in place at local ISPs. To protect the rights of Canal+, the companies must prevent French internet users from using their services to access around 117 pirate domains.

According to the French publication which broke the news, Google attorney Sébastien Proust crunched figures published by government's anti-piracy agency Arcom and concluded that the effect on piracy rates, if any, is likely to be minimal.

Starting with a pool of all users who use alternative DNS for any reason, users of pirate sites – especially sites broadcasting the matches in question – were isolated from the rest. Users of both VPNs and third-party DNS were further excluded from the group since DNS blocking is ineffective against VPNs.

Proust found that the number of users likely to be affected by DNS blocking at Google, Cloudflare, and Cisco, amounts to 0.084% of the total population of French Internet users.

Then, citing a recent survey, which found that only 2% of those who face blocks simply give up and don't find other means of circumvention, he reached an interesting conclusion: "2% of 0.084% is 0.00168% of Internet users! In absolute terms, that would represent a group of around 800 people across all of France!"

In common with other courts which have also been presented with the same arguments, the Paris court said the number of people using alternative DNS to access the sites, and the simplicity of switching DNS, are irrelevant. Canal+ owns the rights to the broadcasts and if it wishes to request a blocking injunction, it has the legal right to do so.

The DNS providers' assertion that their services are not covered by the legislation was also waved aside by the court. Google says it intends to comply with the order. As part of the original matter brought in 2023, it was already required to deindex the domains from search results under the same law.

At least in theory, this means that those who circumvented the original blocks by switching to these alternative DNS services, will be confronted by blocks all over again. But given that circumventing this set of blocks will be as straightforward as circumventing the originals, that raises the question of what measures Canal+ will demand next, and from whom.

So like I said, let's sue copper for having the audacity to indiscriminately carry anyone's electrons. Just as we have in the European Union, regarding whether or not and exactly how and when communications can be encrypted, here we have another instance of a collision between the power of the courts and the power of technology. Technology desperately wants to be and to remain content agnostic. The electrons just don't care. But those who are in the position to impose their will upon the electrons only want them to carry the content they approve of. Google has capitulated and I presume that Cloudflare and Cisco will follow suit.

Before long, DNS is going to become an even greater mess than it already is. And the most annoying part of this is that it's going to be a mess that doesn't even actually solve any real problem since pirates will just switch over to some lesser known well-off-the-map DNS provider that isn't on anyone's radar. And we should also remember that DNS is really only a convenience in the first place. It's a pretty good bet that these pirate content hosting services are using a fixed IP. So just placing an entry into a machine's local HOSTS file will permanently solve the DNS problem by preventing the system from even querying external DNS.

And we should not forget that these piracy streaming sites are being hosted somewhere by someone. THEY are the true culprits and it's they who should be shut down, not honest and well functioning free Internet services offering DNS resolution.

Email @ GRC

During the run-up to today's podcast I almost finished with the code I want to have in place for automating the handling of undeliverable email. So it is still the case that **nothing** has ever been sent **from** GRC, though email is definitely flowing in better than ever. And I expect that to begin sending email later this week.

Among other minor improvements, I'm now pre-checking the domain name portion of user entries by performing a DNS lookup, so typos there are being caught before the user leaves the page. And GRC now supports plus signs in email addresses. So anyone who wishes to filter incoming email by, for example, adding +GRC to the end of their mailbox name may now do so. There's a simple "Delete Account" button on GRC's mail.htm page, so if you enjoy using '+' plus signs you can easily delete your original non-plused email account and create a new one with a plus.

Closing The Loop

Tallis Blalack

Long time Security Now listener and SpinRite owner. What was the program you used to download your email into something you could easily search? You mentioned it on an SN episode a few years ago. My domain host originally offered free email. The cost went to $4.80/year when my email storage went over 2 GB, and I was willing to pay that instead of making the time to reduce my email size. Now, they've moved to email-as-a-service and have increased the cost to $48/year. It's time to back it up and clean it up as I move to a new hosting service. Thanks for all that you do!

This is my periodic opportunity to share one of the best discoveries I've made. The FREE (and wonderful product Tallis is trying to recall is "MailStore Home" which is MailStore's free personal Home edition. As I said, it remains one of my most valued discoveries which I'm happy to recommend without reservation. Through all the years I've been using it, it has never let me down. After installing it, you can aim it at your existing email archive and it will suck it down and build an instantly searchable local database.

The way I have my own personal email system setup is that all incoming mail to me is also cloned into a separate "archive" account, and everything I send is also BCC's (blind carbon copied) into that same "archive" account. That way the archive receives a copy of everything incoming and outgoing. Then, every early morning 3am, Windows Task Scheduler fires up MailStore to retrieve, index and permanently archive everything that has accumulated in that account: https://www.mailstore.com/en/products/mailstore-home/

Astonishingly, all of this is 100% free and I should note that while I've never looked into it, they also have enterprise solutions available for a refreshing one-time purchase, no subscription nonsense. For $260 dollars you get an enterprise-wide email archiving and retrieval solution that integrates with Microsoft 365, Google Workspace and other email providers, even your own.

Eric (who has some first hand info about IT at the New York Times)

Hello Steve, I want to share a comment regarding the NY Times, and a bit of history…..

Around 2010 I was working for a company that provided endpoint security and NYT was our customer. They were stuck on an old unsupported version of our software. Despite all the advances in behavioral-based, machine learning, and non-signature based detection technologies, they insisted on running as "antivirus only". We had countless emails & phone calls documenting our strong recommendation to upgrade and apply the full security stack which they were entitled to.

The recommendations were ignored and they were successfully attacked. They proceeded to publicly name us as their security provider and the technology that failed. Of course we could not go public with a big "I told you so" and were forced to take the black-eye to protect the customer relationship.

So with whatever cyber security incident happened to NYT recently, I do not believe a word they say. I have no doubt the story about the company's IT guy leaving the private GitHub access token exposed is only a cover story for a far worse problem.

Thanks for sharing that, Eric. This really makes one wonder how many of the problems we examine each week are self-imposed. We hear about a critical shortage of qualified IT professionals. But I suppose there's no shortage of unqualified IT wannabes. It would be interesting to know what the real reason was for them not wanting to improve their security when an improvement was being offered and even pushed. From what Eric described, it sounds like it wasn't money since they were already purchasing technology they refused to deploy.

Mark Newton

Steve, I was searching through the show notes. I was thinking you mentioned having a reMarkable 2 or something similar and how much you liked it. It appears there are a couple of different manufacturers out there. I thought you specifically mentioned the reMarkable 2 or perhaps their first version? You would not believe how often the word remarkable applies in the notes from multiple shows. LOL anyway, you sparked my interest. Pricey but looks like it would be helpful.

Leo turned me onto the Remarkable2 stylus-based eInk tablet and I never want to use anything else. Now, in fairness, I did once feel the same way about the Palm Pilot. Extras of them went into the fridge for long-term storage and safekeeping... and they've never come out. Instead I have iPhones and iPads that didn't exist at the time.

So I suppose the lesson is to never say never. But what I can attest to is that through the years since college I kept trying to use some spiffy new technology to replace my use of those wonderful light green quadrule engineering pads with the grid printed on the back side of the paper. Those pads with soft lead mechanical pencil were always the ultimate solution when I was brainstorming or working on a problem. When coding, I'll often make sure I don't make one of those common "off by one" errors by drawing simple diagrams while I run the code in my head.

After 50 years, the Remarkable2 tablet finally replaced those engineering pads for all time.

The Angle of the Dangle

(Dangerous Dangling Domains)

This week's picture of the week presents a step-by-step flow of how Domain Takeover happens. Even though it's entertaining, even though I understood how this happens by the time a listener sent me that fun diagram, untangling the diagram was more work than following an explanation of how this happens. So I want to thank everyone who wrote with their explanations. Since I love my rack of router, firewall, switch and server hardware at Level 3, as I noted at the start last week, I've never so much as glanced at the architecture of cloud-based hosting. Maybe in another 15 years when I can no longer lift a fully loaded server into the rack – but not yet. So the moment a listener mentioned that the trouble was almost certainly with a DNS CNAME record the entire story became clear. And since we have somehow never talked in detail about how DNS CNAME records are used and how badly they can go wrong, I thought this would be the perfect opportunity.

All of this is in reference to last week's topic, the rise and fall of code.microsoft.com, and specifically to how it happened that a malicious actor managed to, however briefly, install their own presence into the 'code' subdomain of microsoft.com. The root of the problem is the potential for **any** organization's DNS records to point to something – anything – that they do not themselves control; because if they do not control the resource that their DNS records point to, the assumption is that someone else might. Which brings us to the question: *"How... could that possibly happen?"* which many of our listeners will recognize as being the close cousin of *"What... could possibly go wrong?"*

The first thing to appreciate is that this general problem has existed since the beginning of the Internet, when DNS was first created. So the issue of anyone's DNS pointing to anything they don't control is not new; and it's never been good. But what is new is the way cloud-hosted resources have evolved, and how, as a result of this, this old problem of DNS misconfiguration has quickly become much greater.

So let's take the fictitious site "BingoBango.com". The people running it decide that they're fed up with running their own website and other services. It's time to move to the cloud. And for the sake of this discussion, they've chosen Microsoft's Azure web hosting service.

When Microsoft created Azure, they obtained the domain azurewebsites.com to act as the root anchor for all of their subscribers' services. So when someone wishes to have Azure host a site and their services, they register with Azure and choose a name for their service. It might be a human readable and meaningful name, like "BingoBango", or it might be just anything that's available like BB123. Regardless of what they choose, their new site is reachable as a subdomain of the Azurewebsites.com domain. So it could be reached at BingoBango.azurewebsites.com or in the case of not caring much about the site's subdomain name, BB123.azurewebsites.com.

But the BingoBango folks already own BingoBango.com, and no one wants to type "bingobango.azurewebsites.com" every time they want to visit the Bingo Bango website. Microsoft was well aware of this, as was Amazon and all of the other cloud providers.

They knew that the names of their services, and its subscriber's subdomains, did not matter because DNS could be used to hide whatever it was.

We're all aware of the way good old standard DNS IP address lookup works. A DNS IPv4 'A' address record is used to turn a DNS name into one or more IPv4 addresses. When a user's client wishes to connect, it uses DNS to look up the domain's IP address and then connects to one of the IP addresses returned. So it's a one-step process. But the success of this one-step process requires that the server's IP address does not change.

If Microsoft could guarantee a fixed IP address for their subscriber's website "bingobango.azurewebsites.com", then standard DNS A-record lookup could be used. Just like the BingoBango people once pointed their BingoBango.com IP address at their own server, they could, instead, point their BingoBango.com IP at the IP address provided by Microsoft. Then visitors to BingoBango.com would simply go to the server at the IP provided by Microsoft.

But DNS provides a better and much more flexible way to achieve the same goal which allows cloud providers total freedom in the way they set up their end. For example, it might be useful to allow different regions of the world to be accessing cloud resources at different IP addresses so that offering a single fixed IP to everyone would be less convenient. So this is made possible by a different type of DNS record known as a CNAME. The 'C' of CNAME stands for canonical.

Whereas a DNS 'A' address record returns one or more IP addresses, a CNAME returns another domain name. In programming we would refer to this as introducing a level of indirection because instead of a domain name pointing directly to an address record, it points to another domain name which then probably points to an address record. And this domain name indirection using DNS CNAME records is the way cloud hosting providers are typically accessed.

Using our example, once the BingoBango people have established their services at bingobango.azurewebsites.com, they replace their DNS's BingoBango.com 'A' record with a CNAME record pointing to bingobango.azurewebsites.com. Now, anytime someone wishes to visit the BingoBango.com website, their web browser will lookup BingoBango.com and will find a CNAME record. That CNAME record tells the browser that the name they were looking up – BingoBango.com – is actually a "canonical name" for the actual hostname, and that CNAME record provides the actual host name, which is bingobango.azurewebsites.com. So THAT is the domain name it should pursue in its quest to find an IP address for its connection. The user's DNS resolver then looks up the IP for the domain bingobango.azurewebsites.com, obtains that IP from the azurewebsites.com DNS servers, and returns that to the user for their connection to the BingoBango.com website.

What's significant here is that two DNS resolutions were required and that the second lookup was for a property located under the azurewebsites.com domain. This allows Microsoft to host their subdomains at whatever IP addresses they choose and affords them the flexibility to change this any time they wish. The BingoBango.com DNS simply refers anyone who asks to the bingobango.azurewebsites.com DNS for its IP.

This CNAME resource creates a very powerful system of DNS pointers, and as we know, with great power comes great responsibility. So it should come as no surprise that things have not

always worked out well. The problem occurs, as happened to Microsoft with their code. microsoft.com subdomain, when an organization deletes the hosted services and domain being pointed to by some other canonical DNS name. At that point the canonical name is said to be left "dangling" because it's pointing to a nonexistent host domain. The bad news is that this opens the way for someone else, who would almost always have to be malicious, to re-register their own cloud service under the same domain name as what was previously deleted.

In the case of the code.microsoft.com subdomain, that subdomain would have had a DNS CNAME record. When that record was retrieved it would have provided the name of the host that had been deleted. So the bad guys would simply register their own host under that CNAME record's name and anyone accessing "code.microsoft.com" would be referred to their probably-malicious services. And note that their servers would appear to any web browsers to be within microsoft.com. This creates additional potential for various browser domain origin trickery.

Three and a half years ago, on November 25th of 2020, "The Daily Swig" cybersecurity news and views newsletter at Portswigger.net, posted a piece under the title *"Rampant CNAME misconfiguration leaves thousands of organizations open to subdomain takeover attacks"*. They wrote:

Security researchers have discovered more than 400,000 subdomains with misconfigured CNAME records, leaving many at risk of malicious takeover as a result.

When websites are externally hosted, the CNAME (Canonical Name) record is used to map their canonical domain and subdomains to the third-party host domain. This means that the canonical, rather than host, domain appears in a browser's address bar.

Pinaki Mondal, of the security firm RedHunt Labs base in India, wrote in blog post that when a cloud hosted website is deleted, but the DNS entry pointing to the resource is retained, attackers can potentially re-register the host, add the organization's subdomain as an alias, and thus control what content is hosted under the original canonical name." Attackers can then serve malicious content to visitors, and potentially intercept internal emails, mount PDF-based clickjacking attacks, hijack users' cookies and sessions by abusing OAuth whitelisting, and abuse cross-origin resource sharing (CORS) to harvest sensitive information from authenticated users.

*Using a tool that conducts mass DNS resolution, RedHunt Labs found more than **424,000** subdomains with misconfigured CNAME records during an automated trawl of 220 million hosts. The number of sites that were abandoned, for example, if they belonged to defunct organization, was unclear due to the additional need to lookup company registries to obtain that information. But by adding HTTP response grabbing, the researchers uncovered evidence that 139 of Alexa's top 1,000 domains may have fallen prey to subdomain takeovers.*

RedHunt Labs identified 33 third-party services that allowed for potential subdomain takeovers.

With nearly 63% of vulnerable DNS records pointing to Shopify, most vulnerable domains belonged to e-commerce operators.

Landing page creator Unbounce accounted for the second highest number of vulnerable domains, at 14%, followed by Heroku (10%), GitHub Pages (4%), and Bigcartel (2%).

Drilling into the data, RedHunt said 'www' was the most frequently vulnerable subdomain, followed by 'shop', 'store', and 'blog'.

Pinaki Mondal pointed out that by suppressing and removing the 'www' and 'm' subdomains from the Chrome browser's address bar from Chrome 69 onwards, Google had inadvertently made it harder for users to realize that they "might be browsing attacker-controlled content".

RedHunt found around 200 "non-functional" .gov site subdomains with misconfigured CNAME records, and one had a 'wildcard' CNAME record, which poses a particularly dangerous security risk. And Mondal noted that prestigious universities owned some of the roughly 1,000 misconfigured .edu subdomains.

The findings show that despite the potentially calamitous impact of subdomain takeovers, many well-resourced large organizations are struggling to comprehensively discover and track their ever-expanding infrastructure.

Mondal also noted that Roblox, Starbucks, and the US Department of Defense are among the organizations to have remediated subdomain takeover flaws through HackerOne in the past year.

The article ended with "The Daily Swig" noting that the year before, in 2019, it had previously reported on subdomain takeover flaws stemming from Windows 8's Live Tiles feature, and the year before that, a misconfigured Microsoft subdomain in 2018.

And four years before this article and RedHunt's research, researchers from the University of Delaware and the College of William and Mary published their research titled *"All Your DNS Records Point to Us – Understanding the Security Threats of Dangling DNS Records"*

Our takeaway for today's podcast is that Microsoft definitely made a serious mistake when they left their "code" subdomain dangling – but they are far from the only organization to have ever done so. In today's increasingly complex IT landscape of overlapping services where such services are increasingly outsourced, DNS can become quite complex and convoluted. And we once talked about how one of the tricks being used to track website visitors is to point a CNAME record of a website to an advertiser's domain as a means of giving them 1st-party cookie-tracking status. That struck us as being ultra slimy at the time, but it's the times we're living through.

So just a heads-up reminder to any of our listeners who have responsibility for the management of DNS for their organization. You likely already know that DNS is not as simple and straight-forward as it once was. It might be worth making sure that you know who any of your organization's CNAME records are pointing to, and that they have a good, ongoing and legitimate reason for being granted such a potentially powerful position since they are sharing your organization's root domain.