Security Now! #876 - 06-21-22
Microsoft's Patchy Patches

This week on Security Now!
We begin this week by answering last week's double-decryption strength puzzler. I then take a look at what's currently known about FIDO2 support in LastPass and Bitwarden. We look at last week's Mozilla announcement of Total Cookie Protection for Firefox (which doesn't appear to be working for me) and invite everyone to test their browsers. DDoS attacks have broken yet another record, another NTLM relay attack has been uncovered in Windows, Apple messed up Safari five years ago, more than a million WordPress sites were recently force-updated, and another high-severity flaw was fixed in a popular JAVA library. Then after sharing a bit of miscellany and some fun closing-the-loop feedback, we look at the awareness the rest of the security industry is sharing regarding the deteriorating quality of Microsoft's security management.

Double Decryption

(Last week's key-strength puzzler)

The question reduces to whether a divide-and-conquer attack can succeed. Way back in 2011, the WPS, WiFi Protected Setup, protocol was found to be vulnerable to this style of attack. The was it was supposed to work was that a user would press a button on their WiFi access point then enter a preset 8-digit PIN into a device they wished to connect to their router. Since the 8th digit was a check digit, only 7 digits were important since the 8th could always be calculated from the first 7. Since 7 digits can have 10 million combinations, brute forcing those 10 million combinations was deemed impractical.

But two researchers, Stefan Viehböck and Craig Heffner, discovered a flaw in the WiFi protocol: All eight digits were not sent at once. The first four digits were sent before the second four, and, worse, the router's behavior would change if the first four were not correct! (How this ever got past the all mighty WiFi Alliance will forever be a mystery... but then again, it was only one of many mistakes they made through the years.) In any event, the fact that the router's behavior would change if the first four were wrong, meant that it wasn't necessary to guess all 7 or 8 digits at a time. It was possible to divide and concur. It was possible to guess just the first 4, of which there are only 10,000 combinations, then, having found the first half, separately brute force the final 3 digits (since the last digit can be calculated from the preceding 7). Since that's 1,000 maximum for the second three, we have a maximum of 11,000 possible guesses, reduced from a previously believed 10 million.

Erik Osterholm's puzzler from last week amounts to the same question: A ciphertext is encrypted with an effective 512-bit key length by first encrypting the original plaintext with a 256-bit key — the first half of the 512-bit key — then by encrypting it again with another 256-bit key — the second half of the 512-bit key.

So now, if we can brute-force decrypt that second encryption in X-time, then bruce-force decrypt the first encryption also in X-time, why isn't this strength just 2X rather than X*X?

And the correct answer, which all of our many listeners who wrote in answered correctly, amounts to whether it's possible to perform the sort of divide-and-conquer attack as was possible and which broke the WPS setup protocol? Remember that the weakness that was exploited by the WPS attack was that there was some affirmative feedback after the first half of the guess was made about whether or not that first half guess was correct... and that's what's missing from Erik's double-encryption thought experiment. Here's how I would phrase it formally:

The result of any encryption by a high-quality cipher, such as AES Rijndael is indistinguishable from entropy. Therefore, the result of the first encryption will be indistinguishable from entropy. So, when following Erik's suggestion, and performing the first decryption, how can the attacker, who is using brute force key guessing, know when their decrypted guess is correct, when both a correct guess and all other incorrect guesses appear equally random. In other words, it is not possible to divide-and-concur. The only way to decrypt the double-encrypted plaintext would be to make a first guess at the outer key. Then the attacker would need to try all possible inner keys, all 2^256 of them to see whether any of them worked. Assuming that none did, all of that work would be discarded, the next outer key would be chosen, and, again, all 2^256 possible inner keys would need to be tried. In other words, only when both the first 256-bit key and the second 256-bit key were simultaneously correctly applied, would the correct plaintext be restored. So this is, indeed, 2^256 times 2^256, which is 2^512 maximum possible brute force guesses needed.

Security News

3rd Party Authenticators

In the aftermath of the Apple's, Google's and Microsoft's announcements of their forthcoming support for FIDO2 and passkeys authentication, we've been talking about what that all means. And I believe that we've settled into exactly the right understanding: It's relatively quick and easy for those three major publishers to add this support into their **clients**. And when they've done so, everyone will be just one software update away from having that client-side technology in their hands. But it’s a bit like creating the first shortwave radio, there’s no one else to talk to yet. So the existence of all of those clients won't be very useful initially. The heavy lift will be getting the millions of individual web servers updated to support the WebAuthn standard at their end, since any use of Apple's, Google's and Microsoft's clients will require that, too.

And I believe that we've also identified that the biggest usability hurdle for the practical use of FIDO2's private passkeys is the need for their dynamic synchronization. And now that the world has sobered up after the intoxicating passkeys announcement parties, others are realizing what we immediately saw as a problem. A story in FastCompany is titled: *“There’s a big problem with Apple and Google’s plans to nix passwords”* and 9To5Mac's headline reads: *“A world without passwords could further lock users into Apple and Google ecosystems”*. Gee, no kidding.

Those stories note that:

> *FIDO’s current proposal has no mechanism for bulk-transferring passkeys between ecosystems. If you want to switch from an Android phone to an iPhone—or vice versa—you won’t be able to easily move all your passkeys over.* [They didn’t mention Windows but we know the same problem will exist there.]
>
> *“We don’t really have a batch export method right now,” says FIDO Alliance executive director Andrew Shikiar. “I think that’s probably a future iteration.”*

Wow. These FIDO guys were really not thinking through the usability angle of all this. “We'd like you all to adopt this half-baked solution today. And we'll worry about exporting your locked-in keys later. The reports also explain:

> *The fear is that if users can easily move all their passkeys between providers, hackers may try to exploit this capability. For now, it’s unclear when or how FIDO might address that problem.*

And then they quote the president of the FIDO Alliance:

> *Sam Srinivas, Google’s product management director for secure authentication—who’s also the*

president of the FIDO Alliance: "It's very hard to do it safely from the get-go, because if we give a mechanism without great care for someone to export all these keys, you know who's going to show up first for that," Srinivas says. "It's not going to be the legitimate user."

In other words, we're going to be quite happy with lock-in and we're going to tell users that it's too dangerous to allow them to move their keys around themselves.
As a cross-vendor user myself, I need Apple and Windows to sync. And I don't see that happening without either a 3rd-party synchronization vendor, or a 3rd-party FIDO2 passkeys client vendor. And that brings us to two password managers which I've been looking into and want to briefly discuss: LastPass and Bitwarden.

As everyone knows, LastPass was previously a many-years sponsor of this podcast and of the TWiT Network. And Bitwarden is currently a sponsor and offers a compelling array of solutions. So the question I had was where do those two fit within this new and evolving era?

As I mentioned last week, I've been annoyed with LastPass because their most recent blog posting, which I was hoping would provide some clarification, left me feeling more confused than I was before. Everything they say feels sort of coy and blurry. Nothing they say just tells us what is going on. Here's an example direct quote from an announcement of an upcoming webinar they'll be hosting in two days, this Thursday. They wrote:

It's time to envision a world without passwords, a world that removes the password-related friction that prevents users from securing and managing their passwords easily and automatically.

True FIDO2-compliant passwordless access to every device, browser, website, and app will take years to develop, but LastPass can get you there sooner.

Join us to learn how LastPass is enabling an end-to-end passwordless experience for the LastPass vault, and all sites stored within. What will this enable you to do?

- *Reduce password related friction for employees*
- *Increase usage and adoption*
- *Set stronger policies and increase security*
- *Fewer lockouts for employees and resets for IT*

Hear from LastPass CTO, Christofer Hoff, as he demonstrates a passwordless login experience, and discusses future plans for FIDO2 authenticators like biometrics and security keys.

As I said, this is like their recent blog posting which doesn't actually say anything. So I'm hopeful that Thursday's webinar might tell us what their plans are, exactly. For anyone who's interested I have a link to the webinar signup in the show notes and I made it this week's GRC shortcut: https://grc.sc/876

https://lp.goto.com/LastPass_Webinar_How_to_Move_Beyond_Passwords_LP.html

Okay. So now let's talk about our current password manager sponsor, Bitwarden.
The first good news is that Bitwarden is already a member of the FIDO Alliance.

I think the problem that any 3rd-party logon system has had, is the chicken and egg problem which makes it difficult to invest in any system which cannot actually be used until it's supported by the world's servers. The flip side of that is that the clear and obvious need for cross-vendor passkey synchronization — which it's now clear, FIDO and Google both just throw their hands up about saying, "yeah, that's a problem" — creates the biggest need and push for 3rd-party passkey managers ever. I wanted to understand where Bitwarden stood, so I did some digging and found some dialog in their community forum:

https://community.bitwarden.com/t/bitwarden-passkey-how-does-bitwarden-fit-into-the-new-microsoft-google-apple-passkey-initiative/41927

Under the title: Bitwarden Passkey (How does Bitwarden fit into the new Microsoft/Google/Apple "Passkey" initiative?)

Q: Microsoft, Google, and Apple have announced support for the FIDO2 passwordless initiative that media are calling "Passkeys". Because Passkeys creates a new key pair for each web site login, there is the issue of moving all these key pairs among devices. I am sure that Google will do that for Android and Chrome, and Apple will do it for their iPhones and Macs, but what about between Android and Apple or Linux? (Not to mention Windows?)

Would Bitwarden be able to support the new Passkeys cross-platform, like it does with current passwords? I want to sync Android to Linux desktop and I will wait for Bitwarden to support this, if the feature will be added.

A: Bitwarden **does** currently support FIDO2 WebAuthn for MFA verification in addition to your master password for **vault** unlocking.

Bitwarden

Two-step Login via FIDO2 WebAuthn | Bitwarden Help & Support 20

This article explains how you can add an extra layer of protection to your Bitwarden Vault by setting up two-step login with a FIDO2 WebAuthn authenticator.

Bitwarden does not support using these "passkeys" to login in lieu of the master password yet, but there is a current similar feature request for this to be supported. Though like most things in adopting to FIDO2 passwordless login fully will take quite a bit of engineering on the backend to integrate.

So, Bitwarden is clearly up to speed with FIDO since they are a member of the Alliance and do already support WebAuthn as a 2-factor method for their vault login. In other words, FIDO2 authenticators can be used to unlock the user's Bitwarden account for traditional automated username and password authentication. What we want, then, is for Bitwarden to also reverse roles to become a FIDO2 authenticator itself, holding the user's FIDO2 passkeys for remote WebAuthn authentication. Their roadmap page has a great deal of discussion about this, so I'm sure it's something they are moving toward.

Firefox: Total Cookie Protection

Last Tuesday, Mozilla's headline read: *"Firefox rolls out Total Cookie Protection by default to all users worldwide"*

The big word that's so easily missed is "default". That's what's changed. Until last Tuesday, "sequestered third party cookies" were optionally available. I had them enabled in my Firefox instances as I imagine many of this podcast's Firefox users also did after we talked about the option. But until now it's been an option. Which means that the majority of Firefox's users would not have had this enabled... since it wasn't the browser's default setting. Now it is.

What's most shocking is that it took us this long to get here because it's such an easy place to get to. This change doesn't disable 3rd-party cookies. It merely divides the single massive global cookie jar into individual per-domain (web engineers would say "same-origin") cookie jars. In that manner, any 3rd party is welcome to set a cookie in anyone's browser. But when that user goes somewhere else, the cookie jar will be switched to a new jar for that new domain. And, again, any 3rd party will be welcome to set their cookie into that jar. But what they will NOT be able to do is to SEE the cookie that they had previously set into the same user's, same browser, when they were visiting that earlier domain. And that simple measure kills cookies whose primary purpose has been cross-domain tracking.

Again, that such a simple measure took so long to happen is astonishing, but it has finally happened.

Now, all that said, it's **not** working for me under Firefox v101.0.1 which appears to be the latest. Chrome is wonderful as I have it set, but Firefox under "Standard" Privacy & Security / Enhanced Tracking Protection is doing nothing. I set it to "Strict" and still nothing. I set it to "Custom" and told it to block cross-site tracking cookies... and still nothing. It was necessary for me to turn off ALL 3rd-party cookies in order to get cross-domain 3rd-party cookie blocking to work.

How do I know? A piece of technology I spent a great deal of time developing many years ago is still functioning. It's GRC's "Cookie Forensics": https://www.grc.com/cookies/forensics.htm

If you want to see for yourself, or test your own browsers' claims and its actual behavior, you can Google *"GRC Cookie Forensics"* and it'll be the first hit that pops up. It allows you to instantly assess the 1st- and 3rd-party cookie handling of any web browser.

As I said, Chrome, as I have it set up, passed with flying colors. But Firefox is currently failing. Perhaps I haven't yet received the "roll-out".

In DDoS News…
We keep breaking DDoS attack records. We're pretty much at the point where our eyes just glaze over at the size of the attacks. And once again, CloudFlare has reported that last week it stopped and mitigated the largest HTTPS DDoS attack on record. That attack weighed in at 26 million requests per second (RPS) and was aimed at a website of a protected Cloudflare customer last week.

Attacks of this size no longer originate from individual compromised hardware devices because they just don't have the speed or connectivity. Instead, most of the attacking IPs were owned by other cloud service providers whose virtual machines and powerful servers had been hijacked to generate the attack.

Since HTTPS query attacks cannot be spoofed, Cloudflare was able to trace the attack back to a powerful botnet of 5,067 devices, each of which generated approximately 5,200 rps (requests per second) at peak.

A Cloudflare spokesperson said that to put the size of this attack in perspective, they had been tracking another much larger — in agent count — but less powerful — in query rate — botnet consisting of over 730,000 devices. Now, rather than letting that number just wash over us, let's stop to think about that for a second. A single identified Botnet consisting of more than 730,000 compromised devices. Holy crap! Nearly three quarters of a million "somethings", all participating in coordinated attacks.

However, the devices **are** apparently light switches and $5 outlet plugs, since that Botnet, while large in number, generated fewer than one million requests per second overall, thus roughly 1.3 requests per second on average per device.

1.3 requests per second per device pretty much classifies the device as a $5 lightswitch or plug. But individually they are still able to generate Internet traffic. The one's I have in my home all phone home to China. Since they live behind a SoHo NAT router — actually two series-connected NAT routers — it's exceedingly unlikely that anyone got into them from the outside. It's FAR more likely that, if they have been up to some mischief, they've been sold into slavery by their original producer. I'll say again that no one should have IoT gadgets attached to their primary home network. It takes deliberate work to set up and maintain a secondary IoT network, but I cannot think of anything more important for residential security. The entity they are phoning home to may not be friendly. And how would you ever know?

The good news is that most recent IoT routers support one or more isolated guest networks. That's what you want to use for those unknowable IoT widgets. This makes the establishment of a secure perimeter far easier and more stable.

Anyway, as for Cloudflare's latest finding, that recent attack was, on average, 4,000 times stronger, due to its use of virtual machines and powerful infrastructure servers. Cloudflare also notes that HTTPS DDoS attacks are more costly to produce than others because these days they require more computational resources, needing to bring up a secure TLS encrypted connection for every attacking query.

The bottom line is, DDoS attacks are no longer survivable unless the target is isolated behind an attack-mitigating bandwidth provider. If you're not, and your organization is being attacked, just declare a holiday and send everyone home until the attack has passed. That's another reality of today's Internet.

MS-DFSNM

As we've all learned, complexity is the sworn enemy of security. In any complex environment consisting of complex interacting components, the addition of another component requires an understanding of that new component's potential interaction with all other components. In the same way that adding each bit to a key doubles that key's total complexity, adding components to a system creates exponential complexity.

And this is one of the biggest dilemmas which we keep seeing that Microsoft has stumbled into. Now we have a newly uncovered type of Windows NTLM — NT LAN Manager — relay attack which has been named DFSCoerce. DFSCoerce leverages the Distributed File System (DFS): Namespace Management Protocol (MS-DFSNM) to seize control of a domain.

When we hear the phrase “a new form of NTLM relay attack” at this point our eyes roll, because you need to wonder just how many different forms of NTLM relay attack can there be if new forms are still being discovered and uncovered in 2022? NTLM relay attacks are a well-known method that exploits Microsoft's original half-baked and never sufficiently robust challenge-response mechanism. We've talked about this so many times before. Rather than simply discarding it decades ago as too broken to fix, they have kept this sickly patient on life support by continually attempting to patch it and wrap it in additional layers of gauze. As a result, today's NTLM relay attacks work the same way today as they did back then: A malicious party sits between clients and servers, intercepts and relays validated authentication requests to gain unauthorized access to network resources, effectively allowing it to gain an initial foothold in Active Directory environments.

Filip Dragovic, the discoverer of this latest wrinkle in the rich NTLM attack surface tweeted: “Spooler service disabled, RPC filters installed to prevent PetitPotam and File Server VSS Agent Service not installed... but you still want to relay [Domain Controller authentication to [Active Directory Certificate Services]? Don't worry MS-DFSNM has your back!”

What is MS-DFSNM? It appears to be another of those things that some random Microsoft engineer added in one of those “Oh! Here's what we need to add to get that done, won’t it be neat?” fits of protocol design 15 years ago back in 2007. It went into Windows and now it can never be removed. For anyone who's curious, it provides yet another remote procedure call (RPC) interface for administering distributed file system configurations.

Just to give everyone a taste for this, here's what Microsoft's first paragraph of their overview of this protocol explains:

The DFS: Namespace Management Protocol [DFSNMP] is one of a collection of protocols that *group shares that are located on different servers by combining various storage media into a single logical namespace. The DFS namespace is a virtual view of the share. When a user views the namespace, the directories and files in it appear to reside on a single share. Users can navigate the namespace without needing to know the server names or shares hosting the data. DFS also provides redundancy of namespace service.*

Okay, sure, that sounds neat. And I guess, 15 years ago when there was nothing better to do then okay let's add that! This is a perfect example of what appears to be complexity for its own sake and nothing could be more antithetical to security. 15 years ago the danger of this should have been appreciated. But it doesn't appear to have been.

In any event, we're stuck with it now. The discovery of this particular DFSCoerce attack follows the related PetitPotam attack which is an abuse Microsoft's Encrypting File System Remote Protocol (MS-EFSRPC) to coerce Windows servers, including domain controllers, into authenticating with a relay under an attacker's control, letting threat actors potentially take over an entire domain. Sound familiar? Yeah. Same exact attack using an entirely different protocol. But don't worry, there's lots more where those came from

The CERT Coordination Center noted, in detailing the attack chain: *"By relaying an NTLM authentication request from a domain controller to the Certificate Authority Web Enrollment or the Certificate Enrollment Web Service on an AD CS system, an attacker can obtain a certificate that can be used to obtain a Ticket Granting Ticket (TGT) from the domain controller."*

And yes, if that makes your head spin, it probably should. Here's my advice: If any of our listeners are offered a job, given responsibility for managing and securing any significant Windows enterprise installation, first, you should start off being single because you will wind up being single. And you should be sure to get a lot of money because you're going to be trading your life and your sanity for that thankless job.

To mitigate NTLM relay attacks, Microsoft recommends enabling protections like Extended Protection for Authentication (EPA), SMB signing, and turning off HTTP on AD servers. Again, mitigations, not cures or solutions. Just wrap it all in some more gauze.

An Apple Safari regression

Although I didn't set out with this goal, this wound up being a pile-on-Microsoft episode. It's not that it's not deserved, but it does feel somewhat redundant. So I'm happy to be able to report that Apple screwed something up too, in a somewhat predictable way...

Google's Project Zero discovered that a security flaw in Apple's Safari was found being exploited in the wild earlier this year. What was interesting about this particular flaw was that it was originally fixed back in 2013, then inadvertently reintroduced in December 2016.

The issue, today tracked as CVE-2022-22620 and bearing a hefty CVSS of 8.8, is another use-after-free vulnerability in WebKit that was being exploited by a piece of specially crafted web content to give its exploiter arbitrary code execution on the machine.

Early in February 2022, Apple shipped patches for the bug across Safari, iOS, iPadOS, and macOS, while acknowledging that it “may have been actively exploited.” Uh huh. And the sun may rise in the morning.

Google Project Zero's Maddie Stone wrote: “In this case, the variant was completely patched when the vulnerability was initially reported in 2013. However, the variant was reintroduced three years later during large refactoring efforts. The vulnerability then continued to exist for 5 years until it was fixed when it was discovered as an in-the-wild 0-day in January 2022.”

Maddie explained that both the October 2016 and the December 2016 commits, one of which reintroduced the original bug from 2013, were very large. The commit in October changed 40 files with 900 additions and 1225 deletions. The commit in December changed 95 files with 1336 additions and 1325 deletions. It seems untenable for any developers or reviewers to understand the security implications of each change in those commits in detail, especially since they're related to lifetime semantics.

So, whatever it was that was happening at the end of 2016, it was apparently a major revamp of some core Safari code. And it appears that you get bit either way. On the one hand, if you leave crappy old code alone under the theory of “if it's not broken don't fix it” you wind up with even older crappy code. But if you bite the bullet to make huge sweeping revamping code-modernizing changes, you get bit by the introduction of brand new bugs which might be the same as some very much older bugs.

Given a choice, I think I'd do what Apple appeared to do. Code really doesn't evolve well. If enough time passes, the assumptions that were once baked into the original code no longer hold true and can really begin to chafe. If the problem that code was written to solve has a strong future, it can be best to scrap a large previous investment, no matter how solid it now is, and start over with all of the benefit of the knowledge acquired through the intervening years.

One Million WordPress sites force-updated
The WordPress security guys at WordFence identified an exploited in-the-wild 0-day bug in a widely-used (more than one million installations) WordPress plug-in called “Ninja Forms”, a customizable contact form builder.

The severity of the bug earned it a CVSS of 9.8 and it affected many versions of Ninja Forms starting with v3.0. WordFence explained that the bug made it possible for unauthenticated attackers to call a limited number of methods in various Ninja Forms classes, including a method that de-serialized user-supplied content. And that resulted in an Object Injection which could allow attackers to execute arbitrary code or delete arbitrary files.

Although the update was supposed to be automatic and forced, any Ninja Forms users listening to this are advised to definitely check their WordPress installations to verify that they are running the latest release of Ninja Forms.

High-Severity RCE in Fastjson Library
And speaking of de-serialization vulnerabilities and attacks, another problem with a very popular JSON library known as “FastJson”, having a CVSS of 8.1, was recently found and fixed.

It was found a patched at the end of May and affected all Java applications that rely on Fastjson versions 1.2.80 or earlier which pass user-controlled data to either the JSON.parse or JSON.parseObject APIs without specifying a specific class to deserialize.

You can already tell that sounds like a bad idea when you hear that a user-provided blog will be de-serialized into a JAVA class without control. In fact, it seems like an inherently bad idea to have anything like that lying around.

Fastjson is a Java library that's used to convert Java Objects into their JSON representation and vice versa. AutoType, the function vulnerable to the flaw, is enabled by default and is designed to specify a custom type when parsing a JSON input that can then be deserialized into an object of the appropriate class.

The guys at JFrog who found the problem said: “However, if the deserialized JSON is user-controlled, parsing it with AutoType enabled can lead to a deserialization security issue, since the attacker can instantiate any class that's available on the Classpath, and feed its constructor arbitrary arguments.” Yikes.

While the project owners previously introduced a safeMode that disables AutoType and started maintaining a blocklist of classes to defend against deserialization flaws, the newly discovered vulnerability gets around these restrictions to result in remote code execution. In other words, this is an inherently frightening capability. And adding a blocklist is a bit like blocking known bad ports on an otherwise open firewall. In other words, you're doing it backwards.

Users of Fastjson are recommended to update to version 1.2.83 or enable safeMode, which turns off the function.

The JFrog guys explained that: “Although a public PoC exploit exists and the potential impact is very high (remote code execution) the conditions for the attack are not trivial (passing untrusted input to specific vulnerable APIs) and most importantly — target-specific research is required to find a suitable gadget class to exploit.” So no widespread exploitation would have occurred.

Miscellany

A few weeks ago I mentioned that I had appeared as a guest of one of our listeners on his own “Trek Profiles” podcast. Last Wednesday John tweeted:

Now comes Episode 66 of Trek Profiles! In this one, we speak with Steve Gibson about his #StarTrek fandom, his history with the show & we endeavor to discover why we all love Trek so much. Plus, you’ll get to meet the mysterious Bunn-Ons. 😱 Get it wherever you get your audio! https://t.co/9TUbrBs3RZ

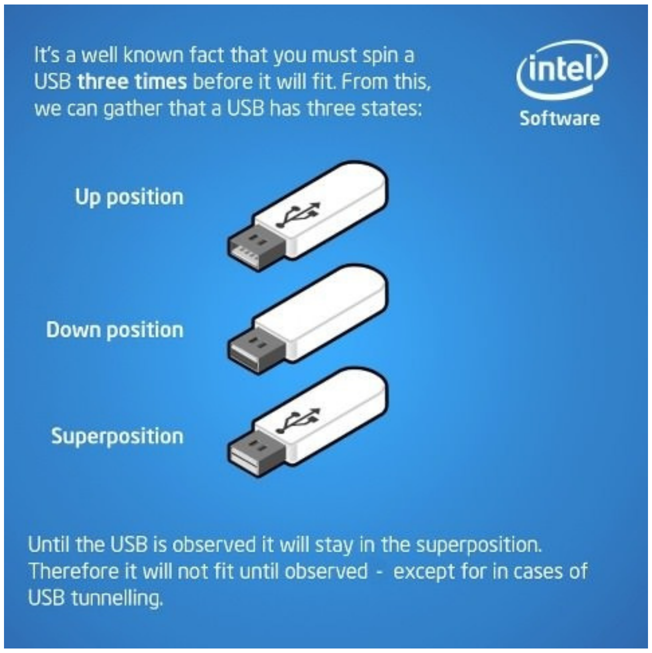
One fun thing that John did was to take the recording he had made of me saying "We are the Bunnons! Surrender your ship or be destroyed!" on the podcast and he reversed it. Although that was 50 years ago, and my timing was a bit off on the "we are the Bunnons part", it was still surprisingly intelligible all these years later.

Anyway, it's the "Trek Profiles" podcast, episode #66, for anyone who might be interested. :)

Closing The Loop

jvstech / @jvs_tech

Hey Steve, I'm sure you've had several others already mention this, but USB type A does _not_ break probability. It is, in fact, in a state of quantum superposition. The connector exists in both the correct and incorrect orientations simultaneously until observed -- at which point, the wave-function collapses and a single orientation manifests. This is why it often takes three tries to plug it in correctly. (Thanks for the amazing podcast, by the way!)


It's a well known fact that you must spin a USB three times before it will fit. From this, we can gather that a USB has three states:
intel Software
Up position
Down position
Superposition
Until the USB is observed it will stay in the superposition. Therefore it will not fit until observed - except for in cases of USB tunnelling.

Brian Tillman / @BrianGTillman

Here's a question I have: suppose I have an account with some service and they decide to support SQRL. How do I get my SQRL identity associated with an existing account?

I wanted to share Brian's question since I'm sure FIDO's passkeys will work the same way: In all cases, an additional method of authenticating is simply added to an account. So you would logon through your traditional means, presumably username and password. Then under your account settings you would choose to add a SQRL or FIDO passkey. You would trigger that authentication query from the website, authenticate with your client, which would provide the website with your specific public key to keep on record.

In the case of FIDO's passkeys that's as far as it goes. But with SQRL, since we thought through the entire problem, we realized that since a system's security is limited by the security of the weakest link, adding a super-secure means of authenticating doesn't actually increase security unless and until you also eliminate the possibility of using all weaker links. That's why SQRL goes further and is able to request that a website disable all non-SQRL authentication. Once a user has become familiar with SQRL and is confident in its use, they're able to set that mode in theirSQRL client, and as they then visit websites, their SQRLclient will be requesting that sites no longer honor their username, password, or any other non-SQRL means of authentication.

Jose C Gomez / @joc85

Hi Steve, I am the owner/host of SQRLOauth that Leo uses on the forum, I had wandered off onto other things. I got alerted by Jeff Arthur (thank you Jeff!) and everything is back up and running. I have added some monitoring tools to keep a better eye on it. I honestly thought nobody was using it, I wish it would get more adoption (SQRL in general) but alas with all this new FIDO, etc. I suspect it isn't going to happen. Nevertheless I have fixed the issues, brought the site back up and even did some bug fixed so the twit community should be back up and running for those that want to use it. Thanks for the heads up, and thanks again to Jeff for reaching out. Cheers!

David Lemire / @dlemire60

Hi Steve. Listen every week and always enjoy the show. But I'd like to suggest you use as careful a definition for IoT as you do for zero day. You often point out how Microsoft's use of the term isn't really appropriate. Well, NIST has a definition for an IoT device in NIST IR 8259 (which has also been adopted into US public law, in the Cybersecurity Improvement Act of 2020). It says such devices have a network interface for interacting with the digital world and a sensor or actuator for interacting with the physical world. I bring this up because you often use routers as examples when you talk about IoT, but by the NIST definition, which is pretty well accepted, a router isn't IoT because it lack an interface to the physical world. I know you like to be precise about things so I thought you'd want to know about the NIST definition.

Well, that's interesting. And I think that we do have a problem here with a weak definition. My original feeling about the use and definition of IoT devices was actually more along the lines of what David cites as NIST's formal definition — things like light switches and plugs and Internet-connected thermostats. But my original definition has been broadened, I think usefully and appropriately, to now include **"unattended devices"** as the primary distinctive feature which determines their "IoT-ness" for the purposes of concerns about security and their abuse. Is a router an IoT device? Increasingly, the Internet security community is adopting that definition. So I don't know what to do. If something like a router is not an IoT device, then it would be nice to have some broadly agreed upon term for the class of Internet-connected and often quite powerful devices which can have firmware flaws allowing them to be remotely subverted.

chuck3000 / @chuck3000
Re: SN 875 - here’s a reason for an electronic pet door :). Apparently this is common in Florida.

Microsoft's Patchy Patches

Today's podcast title was chosen after encountering a number of separate and independent pieces in the tech press, all decrying Microsoft's recent vulnerability and patch handling. Since this has been something that we know I've also been observing and repeatedly noting here, it was somewhat comforting to get a bit of a reality check that my perceptions are not coming out of left field.

Dan Goodin, writing for ArsTechnica, published a piece headlined: *"Botched and silent patches from Microsoft put customers at risk, critics say. Case in point: It took five months and three patches to fix a critical Azure threat."*

And Jonathan Greig writing for The Record separately published: *"Debate rages over Microsoft vulnerability practices after Follina and Azure issues."* Since Jonathan's reporting contained some new information from interviews he conducted across the industry, some from veteran Microsofties, I'll start by sharing some of what Jonathan found:

Microsoft finally released a patch for the much-discussed Follina vulnerability — CVE-2022-30190 — amid fixes for 55 other issues last Tuesday. But Microsoft's initial response to the issue, and several others, has stirred debate among security experts who question Microsoft's recent handling of vulnerabilities.

Microsoft initially claimed Follina "wasn't a security issue" after being sent evidence by the head of advanced persistent threat (APT) hunting organization Shadow Chaser Group. They eventually acknowledged the issue, but several security experts have aired concerns about Microsoft's responses to a number of vulnerability reports.

Last Monday, Amit Yoran, the CEO of the cybersecurity firm Tenable, published a lengthy blog post criticizing Microsoft for its recent response to two disclosed vulnerabilities affecting the Azure Synapse service. In his blog posting, Amit wrote:

"After evaluating the situation, Microsoft decided to silently patch one of the problems, downplaying the risk. It was only after being told that we (Tenable) were going to go public, that their story changed … 89 days after we initially notified them of the vulnerability … when they privately acknowledged the severity of the security issue. To date, Microsoft customers have not been notified. This is a repeated pattern of behavior. Several security companies have written about their vulnerability notification interactions with Microsoft, and Microsoft's dismissive attitude about the risk that vulnerabilities present to their customers."

Yoran went on to say that Microsoft's frequent reticence to notify customers of issues was *"a grossly irresponsible policy."*

In response to questions about Yoran's comments, Microsoft told Jonathan Grieg, reporting for "The Record", that it only assigns CVEs to issues that require customers to take action.

WHAT?!?! So now they're not vulnerabilities if Microsoft handles them secretly? The Microsoft spokesperson said: *"We addressed the issues that Tenable reported to us and no customer action is required."* This apparently is Microsoft's new *"sweep it under the rug"* policy. And, really, when you think about it, isn't this exactly what a behemoth that's answerable to no one would do if it's unable to act responsibly? *"Nothing to see here. What CVE?"*

Aaron Turner, CTO at the security company Vectra, said he understood both sides of the debate as a longtime former Microsoft security team member. Microsoft wants to have the freedom to manage their cloud services the way they see fit, Turner said. He said "I was at Microsoft in the worst of times, from 1999 through 2006 when the company had to go from some of the worst security update management policies to eventually leading the industry in predictability, transparency and one of the best supporters of responsible disclosure." Turner explained that he knows and respects Yoran personally but did not think that Tenable's blog post was constructive. The rules around responsible disclosure do need to be updated, according to Turner, but he noted that both sides "have room for improvement."

And I'll note that if Turner left Microsoft in 2006, when things were going great, according to him, then he will have missed the events of the past 16 years, when things have definitely taken a turn for the worst. In a few minutes I'll be sharing some thoughts from someone who once tested Windows, before Microsoft decided that actual testing was no longer needed.

Turner said there needs to be clearer rules around research into core Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) technologies (in other words, how to deal with Cloud stuff) as well as easier ways for cloud platform operators to provide testing capabilities to researchers and clear responses to responsibly-disclosed vulnerability information. Yeah, that's what Tenable wants.

Several other researchers were less forgiving of Microsoft, pointing out that more than 33% of the vulnerabilities added to the CISA Cybersecurity and Infrastructure Security Agency's list of known exploited bugs... came solely from Microsoft. One third of all. Microsoft had the most vulnerabilities added to the list in every month this year.

Andrew Grotto, former White House Director for Cybersecurity Policy, who is now a cybersecurity professor at Stanford University, argued that Microsoft's market dominance was part of the problem. (Yeah, no kidding. As we know, one of my theories has been that they just don't need to do any since there are no consequences when they do not.)

Grotto explained: *"The data speaks to an outsized representation of Microsoft products having the most critical vulnerabilities. On some level, it may reflect the sheer prevalence of Microsoft products, but it's not like there aren't other vendors whose products are constantly poked and prodded and tested."* No other vendor *"appears with the same frequency and level of severity in terms of vulnerabilities that Microsoft's products seem to"* says Grotto. *"Does the market force Microsoft to remedy this problem or not? What worries me is, right now, there is not a ton of competition, so I'm a bit pessimistic about this trend changing."*

Steven Weber, professor of the Graduate School of Information at UC Berkeley, said procurement is the best way to drive positive changes in security practices. Government procurement practices right now are making the government less secure but also hurting the private markets as well, Weber explained, because it is not creating greater demand for better security. He said: *“It’s important to keep in context that the widespread market penetration of a company’s products is no explanation for why its products are* [also] *the most vulnerable.”* Amen. *“What we ought to be asking is, given that we know, and are shown again and again, that [Microsoft's] products are highly vulnerable, why do they remain so prevalent in the market?”*

And Dan Goodin's reporting added some additional depth to this topic. Reporting in ArsTechnica, Dan observed:

Blame is mounting on Microsoft for what critics say is a lack of transparency and adequate speed when responding to reports of vulnerabilities threatening its customers. Microsoft's latest failing came to light on Tuesday in a post that showed Microsoft taking five months and three patches before successfully fixing a critical vulnerability in Azure.

Orca Security first informed Microsoft in early January of the flaw, which resided in the Synapse Analytics component of the cloud service and also affected the Azure Data Factory. Geet this: **It gave anyone with an Azure account the ability to access the resources of other customers.** From there, Orca explained, an attacker could:

- Gain authorization inside other customer accounts while acting as their Synapse workspace. They wrote we could have accessed even more resources inside a customer’s account depending on the configuration.

- Leak credentials customers stored in their Synapse workspace.

- Communicate with other customers’ integration runtimes. We could leverage this to run remote code (RCE) on any customer’s integration runtimes.

- Take control of the Azure batch pool managing all of the shared integration runtimes. We could run code on every instance.

In other words, this was a horrible and critical vulnerability. Yet Orca said that despite the urgency of the vulnerability, Microsoft responders were slow to grasp its severity. Microsoft botched the first two patches, and it wasn't until just this past Tuesday in June that Microsoft issued an update that entirely fixed the flaw. A timeline provided by Ora shows just how much time and work **it took them** to shepherd and coax Microsoft through the remediation process.

- January 4 – The Orca Security research team disclosed the vulnerability to the Microsoft Security Response Center (MSRC), along with keys and certificates they were able to extract.

- February 19 & March 4 – MSRC requested additional details to aid its investigation. Each time Orca responded the next day.

- Late March – MSRC deployed the initial patch.

- March 30 – Orca was able to bypass the patch. Synapse remained vulnerable.

- March 31 – Azure awards Orca $60,000 for their discovery. Oh, joy. But Azure remains vulnerable

- April 4 (90 days after disclosure) – Orca Security notifies Microsoft that keys and certificates are still valid. Orca still had Synapse management server access.

- April 7 – Orca met with MSRC to clarify the implications of the vulnerability and the required steps to fix it in its entirety.

- April 10 – MSRC patches the first bypass, and finally revokes the Synapse management server certificate. Orca was able to bypass the patch yet again. Synapse remained vulnerable.

- April 15 – MSRC deploys the 3rd patch, fixing the RCE and reported attack vectors.

- May 9 – Both Orca Security and MSRC publish blogs outlining the vulnerability, mitigations, and recommendations for customers.

- End of May – Microsoft deploys more comprehensive tenant isolation including ephemeral instances and scoped tokens for the shared Azure Integration Runtimes.

- In the end... a silent fix, no notification to Microsoft's customers.

And this account from Orca followed 24 hours after the previously mentioned security firm, Tenable, related a similar tale of Microsoft's failing to transparently fix vulnerabilities that also involved Azure Synapse.

And this is wonderful... In a statement, Microsoft officials wrote: *"We are deeply committed to protecting our customers and we believe security is a team sport.* [WHAT?!?! What are you guys smoking up there in Redmond? The security of your proprietary software is a team sport?!?] *We appreciate our partnerships with the security community, which enables our work to protect customers.* [Right, because we don't need to worry about this ourselves, anymore. Those nifty security companies are going to find and fix our problems for us.] *The release of a security update is a balance between quality and timeliness, and we consider the need to minimize customer disruptions while improving protection."*

Wow.

I recalled that a few years ago Microsoft laid off their large and expensive testing teams. So I went looking for, and found, a description of exactly what changed and how, and why today's system has become so badly broken. Martin Brinkman, writing for Ghacks.net had a piece a few years ago titled: *"Former Microsoft Employee explains why bugs in Windows updates increased"*

Have the number of bugs in Windows updates increased in the past couple of years? If so, what is the reason for the increase in bugs? That's the question that former Microsoft Senior SDET (Senior Software Development Engineer In Test) Jerry Berg, recently answered.

Here's Jerry's exuberant 16-minute YouTube video: https://youtu.be/S9kn8_oztsA

Berg worked for 15 years at Microsoft where one of his roles was to design and develop tools and processes to automate testing for the Windows OS. He left the company after Windows 8.1 shipped to the public. Microsoft changed testing processes significantly in the past couple of years. Berg describes how testing was done in the late 2014 early 2015 period, and how Microsoft's testing processes changed since then.

Back in 2014/2015, Microsoft employed many teams that were dedicated to testing the operating system, builds, updates, drivers, and other code. The teams consisted of multiple groups that would run tests and discuss bugs and issues in large daily meetings. Tests were conducted manually by the team and through automated testing, and if tests were passed, would give the okay to integrate the code into Windows.

The teams ran the tests on "real" hardware in labs through automated testing. The machines had different hardware components, e.g. processors, hard drives, video and sound cards, and other components to cover a wide range of system configurations.

Then, Microsoft laid off almost the entire Windows Test team as it moved the focus from three different systems -- Windows, Windows Mobile and Xbox -- to a single system. The company moved most of the testing to virtual machines which meant, according to Berg, that tests were no longer conducted on real and diverse hardware configurations for the most part. They were tested on generic VM's.

Microsoft employees could self-host test releases of Windows which would mean that their machines would also be used for testing purposes. The main idea behind that was to get feedback from in-house Microsoft employees when they encountered issues during their work days. Berg notes that self-hosting is not as widely used anymore as it was before — right, because who wants to be a tester when it means that one's main machine will be crashing.

So now, today, the primary source of testing data, apart from the automated test systems that are in place, comes from outside Microsoft, from Microsoft's users, through Telemetry and Windows Insiders. Windows Insider builds are installed on millions of devices and Microsoft collects Telemetry from all of these devices.

If something crashes, Microsoft gets information about it. Unfortunately, one of the problems associated with the collecting of Telemetry is that most bugs are not caught by it. If something does not work right, Microsoft may not be able to discern the relevant bits from Telemetry data. While it is in theory possible that users report issues, many don't. Additionally, while Insiders may report bugs, it is often the case that necessary information is not supplied to Microsoft which poses huge issues for the engineers tasked with resolving these problems.

Back in 2014/2015, Microsoft's Testing team would be tasked with analyzing bugs and issues, and supplying engineers with the data they required to resolve these. Nowadays, Berg notes, all of that is gone and it's Telemetry that the engineers look at to figure out how to fix these issues and fixes are then pushed to customer devices running Insider Builds again to see if the issue got fixed or if it created new bugs.

One of the main reasons why Microsoft stopped pushing out new feature updates to everyone at once was that issues that were not detected by that process could potentially negatively affect a large number of customers. To avoid total disasters like the Windows 10 version 1809 launch, gradual rollouts were introduced that would prevent feature updates from being delivered via Windows Update to the majority of machines in the early days of the release.

So, Microsoft exchanged the dedicated in-house testing team for remote Telemetry data that it gathers from Insider Builds that it pushes to consumer and business devices, and replaced much of the PCs that it used for testing with virtual environments. All of that has led to an increased number of issues and bugs that customers face on production machines when installing Windows updates or feature updates.

And it appears to have created disconnection throughout various major arteries of Microsoft. I have no doubt that individual Microsoft employees are doing the best they can with what they have. But management appears to have royally screwed the pooch when they decided to disband serious pre-release testing in favor of collecting outside telemetry from Windows Insiders. And we've already talked about what Microsoft plans to do next by further automating the Windows Update system to dynamically back out when things go wrong and to learn from its mistakes. This essentially broadens its Telemetry collection from Windows Insiders to include all other commercial Windows users.

Stepping back to look at the big picture, it becomes clear that what Microsoft has essentially done by disbanding serious internal testing on real hardware, switching to limited testing on virtual machines and outsourcing its software testing first to Windows enthusiasts and then to all Windows users, is to turn us all into their unpaid beta testers. And, as we've seen in example after example, the external security community becomes their thankless security research arm.

So someone hearing this says to me: "Okay, Gibson... What OS are you using?" And we all know the answer to that. I'm sitting in front of Windows. I've always been sitting in front of Windows. And before that I was sitting in front of DOS. I also often sit in front of FreeBSD UNIX which runs several of my most important servers. And I spent a lot of time in front of Debian Linux when I was working to recover that Lenovo laptop's weird-acting Bitlocker-encrypted NVMe drive, the one that I managed to finally get working by hooking it to an external PCIe Thunderbolt port. I used Debian then thanks to the powerful command-line tools that Linux offers.

But the fact remains, I am most comfortable sitting in front of Windows. The tools I prefer using are hosted only there. And the tools I'll be using for SpinRite v7 are **only** hosted there. My failure to preemptively show Microsoft the folly of shipping a consumer OS with userland access to raw sockets taught me that Microsoft had become a blind behemoth. So I have no illusions that we can change Microsoft. But understanding the path and trajectory that Microsoft's policies have put it on, remains valuable.