Security Now! #965 - 03-12-24

Passkeys vs 2FA

This week on Security Now!

What happened with CERT? What headache has VMware been dealing with? What's Microsoft's latest vulnerability disclosure strategy? What's China's "Document 79" and is it any surprise? What long awaited new feature is in version 7.0 of Signal? How is Meta coping with the EU's new Digital Marketing Act that just went into effect? What's the latest on that devastating ransomware attack on Change Healthcare? And after addressing some interesting feedback from our listeners, I want to clarify something about Passkeys that is not at all obvious.

It's certainly a good thing that bag of dirt was labeled with a "ground" symbol, or the dirt's purpose might have been unclear.

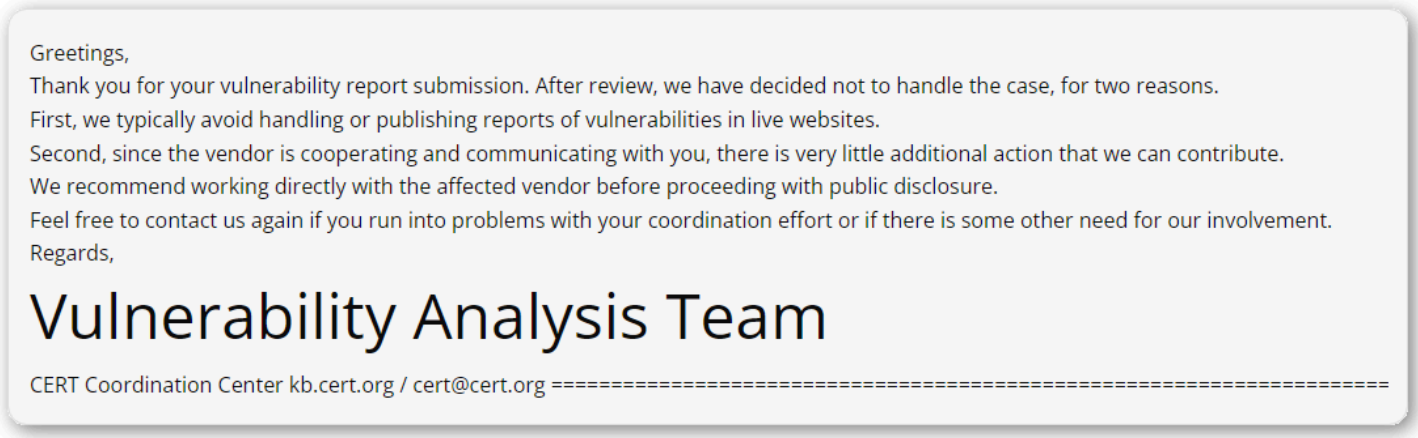
Security News

CERT wasn't much help

Recall from last week our listener who discovered a serious flaw in the website of a major enterprise with hundreds of millions of users. The problem was that they were using a very outdated version of the nginx web server that contained a well known critical remote code execution vulnerability for which working Python proof of concept code was readily available.

After giving the company 120 days to explore and correct the problem, and then prompting them several more times, they finally said that their security team had judged that this was not an issue, but that if our listener promised to remain silent about this, they would pay a $350 bounty.

He sent me a Tweet asking what I thought his next steps should be and I recommended that he file a vulnerability discovery report through CISA, which would be handled by US CERT. Well, that also hit a dead end. CERT replied:



Greetings,
Thank you for your vulnerability report submission. After review, we have decided not to handle the case, for two reasons.
First, we typically avoid handling or publishing reports of vulnerabilities in live websites.
Second, since the vendor is cooperating and communicating with you, there is very little additional action that we can contribute.
We recommend working directly with the affected vendor before proceeding with public disclosure.
Feel free to contact us again if you run into problems with your coordination effort or if there is some other need for our involvement.
Regards,

Vulnerability Analysis Team

CERT Coordination Center kb.cert.org / cert@cert.org ==========

Well, that's disappointing. They say *"we typically avoid handling vulnerabilities in live websites."* So they limit their handling of vulnerabilities to dead websites? That doesn't inspire confidence. Our listener's correspondence, which he had forwarded to CERT with his report made clear that the company had no intention of doing anything further. So CERT was just passing the buck back to someone who had already demonstrated that he had no leverage. In my reply to him, since he had never disclosed his discovery publicly, I suggested that since he had done all he could reasonably do, he should take the money as compensation for his trouble and leave whatever happens up to fate. If the company eventually gets bitten it will only be their fault.

VMware needs immediate patching

One week ago, on March 5th, Broadcom, VMware's parent company issued a security advisory, VMSA-2024-0006, which addresses security vulnerabilities discovered in VMware ESXi, VMware Workstation Pro & Player, and VMware Fusion. VMware's ubiquitous USB virtualization drivers contain four critical flaws. And they are so bad that VMware has issued patches for previous end-of-life releases. An attacker who has privileged access in a guest OS VM, so a root or admin user, may exploit these vulnerabilities to break out of the virtualization sandbox to access the VMware hypervisor.

Patching VMware immediately is the optimal solution. But if anything prevents that from happening, and if your environment is at risk because it includes untrusted VM users, the removal of VMware's USB controllers from virtual machines will prevent exploitation until patches can be applied. VMware said that the prospect of a hypervisor escape warranted an immediate response under the company's IT Infrastructure Library – ITIL. They said: "In ITIL terms, this situation qualifies as an ***emergency change***, necessitating prompt action from your organization."

Both the UHCI and XHCI USB controller drivers contain exploitable use-after-free vulnerabilities, each having a maximum severity range of 9.3 for Workstation/Fusion and a base score of 8.4 for ESXi. There's also an information disclosure vulnerability in the UHCI USB controller with a maximum CVSSv3 base score of 7.1. Someone with administrative access to a virtual machine can exploit it to leak memory from the vmx process. I've included a link in the show notes to VMware's FAQ about this for anyone who might be affected:
https://core.vmware.com/resource/vmsa-2024-0006-questions-answers

Midnight Blizzard – don't get sprayed by passwords!
Unfortunately, Microsoft's networks, and apparently their customers, remain under attack by the Russian Midnight Blizzard group. This is the group that successfully infiltrated some of Microsoft's top corporate executives by attacking the security of a system that had not been updated to current security and authentication standards. The sad thing is that Microsoft has apparently now taken to dropping their P.R. Bombs late on Fridays as happened again this past Friday. The Risky Business security newsletter explained things by writing:

> *Microsoft says that Russian state-sponsored hackers successfully gained access to some of its internal systems and source code repositories. The intrusions are the latest part of a security breach that began in November of last year and which Microsoft first disclosed in mid-January.*
>
> *Initially, the company said hackers breached corporate email servers and stole inboxes from the company's senior leadership, legal, and cybersecurity teams. In an update on the same incident posted late Friday afternoon**—as is the practice of every respectable corporation**—Microsoft says it found new evidence over the past weeks that the Russian hackers were now weaponizing the stolen information.*
>
> *The Redmond-based giant initially attributed the attack to Midnight Blizzard, a group also known as Nobelium, one of the cyber units inside Russia's Foreign Intelligence Service (SVR). Microsoft says that since exposing the intrusion, Midnight Blizzard has increased its activity against its systems. Per the company's blog post, Midnight Blizzard password sprays increased in February **10 times** compared to the "already large volume [it] saw in January 2024."*
>
> *Furthermore, if we read between the lines, the group is now also targeting Microsoft customers. While Microsoft's legalese makes it clear that no customer-facing systems were compromised, the company weaselly confirmed that customers have been "targeted". Emails stolen by the Russian group also contained infrastructure secrets (aka secret tokens, passwords, API keys, etc.), which Midnight Blizzard has been seen attempting to use.*

It's not good that Russians managed to compromise Microsoft executives' eMail and that the information they stole is now being leveraged to facilitate additional intrusions, including the

exfiltration of Microsoft's source code repositories. That's not good for anyone. But as we know, I always make a clear distinction between mistakes and policies. Mistakes happen. Certainly a big one happened here. But the new policy that caught my eye was that, unfortunately, we are now seeing a pattern develop of late Friday afternoon disclosures of information Microsoft hopes will be picked up and published when fewer people are paying attention. Oh well.

China is quietly "de-American'ing" their networks

The official name for Beijing's secret directive is "Document 79" but it is informally known as the "Delete A" order where the "A" is understood to stand for America. The secret directive was issued several years ago, in September of 2022. It is designed to remove American and other non-Chinese hardware and software from its critical sectors. The directive mandates that state-owned companies replace all US and other non-Chinese equipment and software in their IT systems by 2027. Affected companies include Cisco, IBM, Dell, Microsoft and Oracle.

I ask, who can blame them? With tensions on the rise between the US and China, this only makes sense. The US has done the same thing for the manufacturers of Chinese-made security cameras. And we know that Russia is working to remove American technological influence from inside its borders. The problem for all parties is that today's systems are so complex that Trojan capabilities are readily hidden and can be made impossible to find. No one doubts the influence that Chinese intelligence services are able to exert over the design of Chinese equipment. And there has long been some question about how much influence US intelligence services might have over the installation of such backdoors in proprietary software. It's unfortunate to be seeing this pulling apart since it is ultimately much less efficient. But it's hardly surprising.

Signal's Version 7.0, now in beta

Signals reliance upon physical phone numbers for identifying communicating parties has been a longstanding annoyance as well as a concern for privacy and security researchers who have long asked the company to switch from phone numbers to usernames to protect users' identities. The just announced v7 of Signal will add the creation of temporary username aliases. Here's how Signal's announcement explains it:

> *Signal's mission and sole focus is private communication. For years, Signal has kept your messages private, your profile information (like your name and profile photo) private, your contacts private, and your groups private – among much else. Now we're taking that one step further, by making your phone number on Signal more private. Here's how:*
>
> *If you use Signal, your phone number will no longer be visible to everyone you chat with by default. People who already have your number saved in their phone's contacts **will** still see your phone number since they already know it.*
>
> *If you don't want to hand out your phone number to chat with someone on Signal, you can now create a unique username that you can use instead (you will still need a phone number to sign up for Signal). Note that a username is not the profile name that's displayed in chats, it's not a permanent handle, and not visible to the people you are chatting with in Signal. A username is simply a way to initiate contact on Signal without sharing your phone number.*

If you don't want people to be able to find you by searching for your phone number on Signal, you can now enable a new, optional privacy setting. This means that unless people have your exact unique username, they will not be able to start a conversation, or even know that you have a Signal account – even if they do have your phone number.

These options are in beta, and will be rolling out to everyone in the coming weeks and once these features reach everyone, both you and the people you are chatting with on Signal will need to be using the most updated version of the app to take advantage of them.

Also, all of this is optional. While we changed the default to hide your phone number from people who don't have it saved in their phone's contacts, you can change this setting. You are not required to create a username and you have full control over whether you want people to be able to find you by your phone number or not. Whatever choices work for you and your friends, you'll still be able to communicate with your connections in Signal, past and present.

So that seems all good. They've changed to hiding phone numbers by default and they've added an optional textual username as a phone number alias. Signal knows the phone number behind the alias but users do not. That's pretty slick. Their blog posting about this contains a great deal more information. A full user guide. I have it in this week's show notes since it might be of interest to anyone who uses Signal often: Phone Number Privacy & Usernames.

Meta, WhatsApp and Messenger -meets- the EU's DMA

Last Thursday the 7th, the European Union's Digital Markets Act (DMA) went into force. Among many important features, it requires interoperability among instant messaging systems. Here's what Meta recently explained about their plans. It amounts to: If you want to talk to us, you gotta use the Signal protocol... but we'll meet you halfway by making it possible for you to connect with our previously closed servers. Meta wrote:

To comply with a new EU law, the Digital Markets Act (DMA), which comes into force on March 7th, we've made major changes to WhatsApp and Messenger to enable interoperability with third-party messaging services. We're sharing how we enabled third-party interoperability (interop) while maintaining end-to-end encryption (E2EE) and other privacy guarantees in our services as far as possible.

On March 7th, a new EU law, the Digital Markets Act (DMA), comes into force. One of its requirements is that designated messaging services must let third-party messaging services become interoperable, provided the third-party meets a series of eligibility, including technical and security requirements.

This allows users of third-party providers who choose to enable interoperability (interop) to send and receive messages with opted-in users of either Messenger or WhatsApp – both designated by the European Commission (EC) as being required to independently provide interoperability to third-party messaging services.

For nearly two years our team has been working with the European Commission to implement interop in a way that meets the requirements of the law and maximizes the security, privacy and safety of users. Interoperability is a technical challenge – even when focused on the basic functionalities as required by the DMA. In year one, the requirement is for 1:1 text messaging

between individual users and the sharing of images, voice messages, videos, and other attached files between individual end users. In the future, requirements expand to group functionality and calling.

To interoperate, third-party providers will sign an agreement with Messenger and/or WhatsApp and we'll work together to enable interoperability. Today we're publishing the WhatsApp Reference Offer for third-party providers which will outline what will be required to interoperate with the service. The Reference Offer for Messenger will follow in due course.

While Meta must be ready to enable interoperability with other services within three months of receiving a request, it may take longer before the functionality is ready for public use. We wanted to take this opportunity to set out the technical infrastructure and thinking that sits behind our interop solution.

Our approach to compliance with the DMA is centered around preserving privacy and security for users as far as is possible. The DMA quite rightly makes it a legal requirement that we should not weaken security provided to Meta's own users.

The approach we have taken in terms of implementing interoperability is the best way of meeting DMA requirements, whilst also creating a viable approach for the third-party providers interested in becoming interoperable with Meta and maximizing user security and privacy.

First, we need to protect the underlying security that keeps communication on Meta E2EE messaging apps secure: the encryption protocol. WhatsApp and Messenger both use the tried and tested Signal Protocol as a foundational piece for their encryption.

Messenger is still rolling out E2EE by default for personal communication, but on WhatsApp, this default has been the case since 2016. In both cases, we are using the Signal Protocol as the foundation for these E2EE communications, as it represents the current gold standard for E2EE chats.

In order to maximize user security, we would prefer third-party providers to use the Signal Protocol. Since this has to work for everyone however, we will allow third-party providers to use a compatible protocol if they are able to demonstrate it offers the same security guarantees as Signal.

To send messages, the third-party providers have to construct message protobuf structures which are then encrypted using the Signal Protocol and then packaged into message stanzas in eXtensible Markup Language (XML).

Meta servers push messages to connected clients over a persistent connection. Third-party servers are responsible for hosting any media files their client applications send to Meta clients (such as image or video files). After receiving a media message, Meta clients will subsequently download the encrypted media from the third-party messaging servers using a Meta proxy service.

It's important to note that the E2EE promise Meta provides to users of our messaging services requires us to control both the sending and receiving clients. This allows us to ensure that only the sender and the intended recipient(s) can see what has been sent, and that no one can listen to your conversation without both parties knowing.

While we have built a secure solution for interop that uses the Signal Protocol encryption to

protect messages in transit, without ownership of both clients (endpoints) we cannot guarantee what a third-party provider does with sent or received messages, and we therefore cannot make the same promise.

Whatsapp & Messenger Messaging Interoperability in the EU

The posting then goes on in significant detail to explain how Meta is opening its messaging platforms to other providers. I have a link in the show notes for anyone who may be interested. But as we saw, it's a good approach. Signal is open. Don't reinvent the wheel or you may be unable to prove that what you do is equal to Signal. And unless you can prove that we're off the hook.

The Change Healthcare cyberattack

Nearly three weeks ago, on Wednesday February 21st, the American company Change Healthcare, a division of UnitedHealth Group, was hit by a ransomware attack that was devastating by any measure. That cyberattack shut down the largest healthcare payment system in the United States.

This past Sunday, the US Department and Health and Human Services addressed an open letter to "Health Care Leaders", writing:

As you know, last month Change Healthcare was the target of a cyberattack that has had significant impacts on much of the nation's health care system. The effects of this attack are far-reaching; Change Healthcare, owned by UnitedHealth Group (UHG), processes 15 billion health care transactions annually and is involved in one of every three patient records. The attack has impacted payments to hospitals, physicians, pharmacists, and other health care providers across the country. Many of these providers are concerned about their ability to offer care in the absence of timely payments, but providers persist despite the need for numerous onerous workarounds and cash flow uncertainty.

The day following the attack, on February 22nd, UnitedHealth Group filed a notice with the US Securities and Exchange Commission stating that "a suspected nation-state associated cybersecurity threat actor" had gained access to Change Healthcare's networks. Following that UHG filing, CVS Health, Walgreens, Publix, GoodRX, and BlueCross BlueShield of Montana reported disruptions in insurance claims. The cyberattack affected family-owned pharmacies and military pharmacies, including the Naval Hospital at Camp Pendleton, the Healthcare company AthenaHealth was affected, as were countless others.

One week later, on the 29th, UHG confirmed that the ransomware attack was "perpetrated by a cybercrime threat actor who represented itself to Change Healthcare as ALPHV/Blackcat." In the same update, the company stated that it was "working closely with law enforcement and leading third-party consultants, Mandiant and Palo Alto Networks" to address the matter. And then, four days later, eight days ago on March 4th, Reuters reported that a bitcoin payment equivalent to nearly $22 million USD was made to a cryptocurrency wallet "associated with ALPHV." UnitedHealth has not commented on the payment, instead stating that the organization was "focused on the investigation and the recovery." – right! Apparently to the tune of $22 million US dollars. On the same day, a reporter at Wired stated that the transaction looked "very much like

a large ransom payment."

What's transpired since then is a bit interesting, since ALPHV/Blackcat is a ransomware as a service group. This of course means that they provide the software and back-end infrastructure while their affiliates perpetrate the attacks and in turn receive the lion's share, in this case 70%, of any ransoms paid. However, in this instance it appears that ALPHV/Blackcat is not eager to part with that 70%, which amounts to a cool $15.4 million. So they are claiming that they've shut down and disbanded. Nice timing.

Exactly one week ago, the HIPPA Journal posted some interesting information. They wrote:

The ALPHV/Blackcat ransomware group appears to have shut down its ransomware-as-a-service (RaaS) operation, indicating there may be an imminent rebrand. The group claims to have shut down its servers, its ransomware negotiation sites are offline, and a spokesperson for the group posted a message, "Everything is off, we decide." A status message of "GG" was later added and ALPHV/Blackcat claimed that their operation was shut down by law enforcement and said it would be selling its source code.

However, security experts disagree and say there is clear evidence that this is an exit scam, where the group refuses to pay affiliates their cut of the ransom payments and pockets 100% of the funds. ALPHV/Blackcat is a ransomware-as-a-service operation where affiliates are used to conduct attacks and are paid a percentage of the ransoms they generate. Affiliates typically receive around 70% of any ransoms they generate and the ransomware group takes the rest.

After the earlier disruption of the Blackcat operation by law enforcement in December of last year, Blackcat has been trying to recruit new affiliates and has offered some affiliates an even bigger cut of the ransom. An exit scam is the logical way to wind up the operation and there would likely be few repercussions, other than making it more difficult to recruit affiliates if the group rebrands.

It is not unusual for a ransomware group to shut down operations and rebrand after a major attack, and ALPHV/Blackcat likely has done this before. ALPHV/Blackcat is believed to be a rebrand of the BlackMatter ransomware operation, which was a rebrand of DarkSide. DarkSide was the ransomware group behind the attack on Colonial Pipeline in 2021 that disrupted fuel supplies on the Eastern Seaboard of the United States. Shortly after the attack, the group lost access to its servers, which they claimed was due to the actions of their hosting company. They also claimed that funds had been transferred from their accounts and suggested they were seized by law enforcement. BlackMatter ransomware only lasted for around 4 months before it was shut down, with the group rebranding in February 2022 as ALPHV/Blackcat.

On March 3, 2024, an affiliate with the moniker Notchy posted a message on Ramp Forum claiming they were responsible for the attack on Change Healthcare. The post was found by a threat researcher at Recorded Future. Notchy claimed they were a long-time affiliate of the ALPHV/Blackcat operation, had the "affiliate plus" status granting them a larger piece of the pie, and that they had been scammed out of their share of the $22 million ransom payment.

*They claimed that Optum paid a 350 Bitcoin ransom to have the stolen data deleted and to obtain the decryption key. Notchy shared the payment address which shows a $22 million payment **had** been made to the wallet address and the funds have since been withdrawn.*

The wallet has been tied to ALPHV/Blackcat as it received payments for previous ransomware attacks that have been attributed to the group.

Notchy claimed ALPHV/Blackcat suspended their account following the attack and had been delaying payment before the funds were transferred to Blackcat accounts. Notchy said that Optum paid to have the data deleted but they have a copy of 6TB of data stolen in the attack. Notchy claims the data includes sensitive information from Medicare, Tricare, CVS-CareMark, Loomis, Davis Vision, Health Net, MetLife, Teachers Health Trust, tens of insurance companies, and others. The post finishes with a warning to other affiliates that they should stop working with ALPHV/Blackcat. It is unclear what Notchy plans to do with the stolen data and whether they will attempt to extort Change Healthcare or try to sell or monetize the data.

Fabian Wosar, Emsisoft's CTO is convinced this is an exit scam. After checking the source code of the law enforcement takedown notice, he said it is clear that Blackcat has recycled it from December's takedown notice. Fabian Tweeted: "There is absolutely zero reason why law enforcement would just put a saved version of the takedown notice up during a seizure instead of a new takedown notice." He also reached out to contacts at Europol and the NCA who said they had no involvement in any recent takedown. Currently, neither Change Healthcare nor its parent company UnitedHealth have confirmed if they paid the ransom and issued a statement saying they are currently focused on the investigation.

So this is all a big mess. It appears that the Blackcat gang has made off with Optum's $22 million dollars, the Notchy affiliate didn't get the $15.4 million or more that they feel they deserve, Optum got neither the decryption keys nor the deletion of their 6TB of data and no one but the Blackcat guys are smiling. Since rebranding and returning to the ransomware as a service business may be impossible after taking their affiliate's money, and since $22 million is a nice piece of change, they may just go find a nice beach somewhere to lie on.

Meanwhile, here in the States, the inevitable class action lawsuits have been filed due to the loss of patient care health records. At last count at least five lawsuits are now underway.

SpinRite

Everything continues to proceed well with SpinRite. The limitations v6.1 has – of being unable to boot on UEFI-only systems and its lack of native high-performance support for drives attached by USB and NVMe media are as annoying for SpinRite's users as they are for me. So I haven't slowed down at all. I'm working to get v6.1 solidified so that I can get started as soon as possible on v7.0, which will not have any such limitations. That said, I do still want to solve any remaining problems that I can, especially when such a solution will be just as useful for tomorrow's SpinRite 7 as for today's v6.1. And that's the case for the forthcoming feature I worked on all last week. It's something I've mentioned before which I've always planned to do, and that's to add the capability for Linux users, who don't have ready access to Windows, to directly download an image file that can be transferred to any USB drive to boot their licensed copy of SpinRite.

For those who don't know, the single SpinRite executable (about 280K) is both a Windows and a DOS application. When it's run from DOS it is SpinRite. But when that same program is run from Windows it presents a familiar Windows user interface which allows its owner to easily create bootable media which contains itself. So it can either format and prepare a diskette (which there's not much demand for anymore, but the code was all there from v6.0 so I left it alone). Or it can create an optical disc ISO image file for burning to a disc or loading with an ISO boot utility. It can also create a raw IMG file or prepare a USB thumb drive. 20 years ago, back in 2004 when I first created this code, the dependence upon Windows provided a comprehensive solution because Linux was still mostly a curiosity and hadn't yet matured into the strong alternative OS it has since become. Today, there are many Linux users who would like to use SpinRite but who don't have ready access to Windows in order to run SpinRite's boot prep. And this need for non-Windows boot preparation will continue with SpinRite 7 and beyond.

The approach I've developed for use under Windows, which is used by InitDisk, ReadSpeed and SpinRite, is to start the application, then have its user insert their chosen USB thumb drive while the application watches the machine for the appearance of any new USB drive. This bypasses the need to specify a drive letter, it works with unformatted drives, drive with foreign file systems, and makes it **very** difficult for the user to format the wrong drive, which was my primary motivation for developing this user-friendly foolproof approach. Unfortunately, there's a downside, it uses low-level Windows USB monitoring which has not been implemented in WINE, the Windows emulator for Linux.

So, for Linux users, a ready-to-boot image file is the way to go and I'm in the process of putting the final pieces of that facility together. It should be finished and available later this week.

Closing the Loop

John Robinette / @jrobinette

Hey Steve, I'm sure I'm not the only one to send you this note about Telegram after listening to SN#964. There has been much chatter about the protocol Telegram uses for end-to-end encryption, but it is a common misunderstanding that they use this by default.

*Telegram's default uses *only* TLS to protect the connection between your device and their servers (https://telegram.org/faq#q-so-how-do-you-encrypt-data) and does not provide any end-to-end protection. They have an additional feature "Secret Chats", that uses end-to-end encryption on the client device (https://telegram.org/faq#q-how-are-secret-chats-different). It is not possible to use end-to-end encryption in group chats, and when used for one-on-one chats it limits the conversation to a specific device. Based on my anecdotal experience using Telegram with a few friends, most people either do not know about "Secret Chats" or do not use it.*

*I found a post from 2017 that explains Telegram's reasoning for this: (https://telegra.ph/Why-Isnt-Telegram-End-to-End-Encrypted-by-Default-08-14) The TL;DR is: because other apps (ie: WhatsApp) allow you to make unencrypted backups, actual end-to-end encryption isn't worth being on by default. So they don't turn it on by default and claim they are *more* secure because of how they store backups. Anyway, it's not just Apple marketing speak that Telegram isn't end-to-end encrypted. It's by design from Telegram. Thanks!*

I want to thank John and several other of our listeners who have actually used Telegram and who shared their experiences. What they showed was that unlike the way I presumed it would be used when I talked about it last week, Telegram really is probably mostly used in its insecure messaging mode since enabling it is **not** a global setting; it must be explicitly enabled on a chat-by-chat basis. That makes its use actively hostile to end-to-end encryption, which is certainly not what someone hoping for privacy wants. And its fundamental inability to provide end-to-end protection for multi-party group chats strongly suggests that it is being left behind. I presume that they're aware of this and are hopefully working hard behind the scenes to bring Telegram up to speed, since otherwise it's just going to become an historical footnote. Given these facts, I reverse myself and agree with Apple's placing Telegram at Level 0 where it certainly belongs.

3n0m41y / @3n0m41y

Steve, KeepassXC latest release is also supporting passkeys now! Great news! https://keepassxc.org/docs/KeePassXC_UserGuide#_passkeys

I'm not a Keepass user, so I don't know what communication they provide to their users, and I wanted to pass along that welcome news. Back when Passkeys first appeared, supported only by Apple, Google and Microsoft, they each created their own individual and well-isolated walled gardens to manage passkeys for their devices. At that time we hoped that our password managers would be stepping up and getting in on the act because they would be able to offer fully cross-platform passkey synchronization. And as we know, Bitwarden, a TWiT network sponsor, has done so. And given competitive pressures it will soon become incumbent upon any password manager to also offer passkeys support. We'll be talking more about that that means

when we get to this week's main topic.

Q3RRV7sRmg2be / @Q3RRV7sRmg2be

Episode 964: I am on the go and don't have time to write an eloquently worded message like the ones you read on the show, so feel free to simply summarize this if you find it relevant.

in 964 you mentioned feedback from a listener who mentioned the Taco Bell app using email passwords "for convenience". Before my main comment, I'd like to mention that they are not the only ones doing this. I suspect the most popular service out there today that does so is Substack, the blogging service that has exploded in popularity since early/mid COVID. When I signed up there for the first time I wasn't aware and the whole process was very confusing and counterintuitive given my learned user/password behavior. Beyond being confusing, it truly is much more of a pain for those of us who use browser based password managers, which have made the traditional process rather seamless.

Right. The more we explore this topic, given that passwords are entirely optional when every site includes an "I forgot my password" recovery link, I think that viewing passwords as "login accelerators" which are handled by our password managers is quite apropos.

Anyways, my main comment is that, it may very well be possible that these implementations are not for user convenience, they are for liability. By requiring email to login, the host company hoists account breach liability off of their own shoulders onto the email providers. Your account got hacked? That's a gmail breach, not our problem!

On a side note, you also mentioned in the episode a listener in the financial sector who commented on password length, complexity, and rules being limited by old legacy mainframes. I just wanted to share that I worked for a federal agency about a decade ago, and thought I discovered a bug when I realized I had just been let into my sole enterprise account despite missing a character on the end of my password. I tested again and discovered the system was only checking the first 8 characters of the password I thought I had. I grabbed a fellow developer to show them thinking I'd blown the lid off something big, and he shrugged and said "yeah it only checks the first 8 characters". I was stunned.

Thanks for the wonderful content. Cheers.
P.S. Yes, this user account name was generated by my BitWarden password generator.

We first took up the subject of eMail-only login with our "Unintended Consequences" episode, since it was becoming clear that websites were planning to react to the loss of 3rd-party tracking with the establishment of 1st-party identity relationships with their visitors. And in every instance the company's so-called "privacy policy" clearly stated that any and all information provided by their visitors – including their name and their eMail address – not only could be, but would be, shared with their "business partners" – which we know are the advertising networks providing the advertising for their site.

But I thought this listener's idea about limiting liability was interesting, and it had not occurred to me. When you think about it, what's the benefit to a website of holding onto the hashes of all of its user's passwords? The only benefit to the **site** is that passwords allow its users to login

more quickly and easily when and if they're using a password manager. The problem is, the site cannot know whether the user is using a password manager. How big an issue is that? Today, only one out of every three people (34%) **are** using a password manager. And while this is a big improvement over just two years ago in 2022 when the figure was about one in five (21%), this means that today, two out of every three Internet users are NOT using password management. And we know this suggests that the quality of those 2/3rds of all passwords may not be very high. They may no longer be using "monkey123" but their chosen passwords are likely not much better.

Now look at what happened with the 23andMe disaster. A shocking number of 23andMe user accounts were apparently compromised using some form of password spray. That could never have succeeded if 23andMe users were logging in with highly complex unique-per-site passwords. But with 2/3rds of Internet users still, today, not using a password manager, that suggests that 2/3rds of 23andMe users were not well protected from the abuse of their poor password hygiene. If you're not someone who understands the value of password management, "monkey123" looks pretty good and it sure is easy to remember!

My point is: The quite successful attack on 23andMe demonstrates that the use of traditional username and password login creates a single point of failure which facilitates automated attack. A widely distributed network of bots can create authentic appearing web sessions and attempt to login as legitimate users – over and over and over under cover of darkness – succeeding eventually and incrementally – **with no involvement on any user's part**. And that makes a crucial difference. By comparison, if a user's eMail inbox were to be flooded with 23andMe login requests, every such user – and quite soon 23andMe – would know that someone or something was up to no good. But would an attacker even bother? Doesn't the eMail loop completely thwart such attacks? Logins are no longer autonomous. And unless the attacker obtains access to every user's eMail, the attacker cannot login using the user authentication data contained in the eMail. This eliminates all generic remote password spraying brute force attacks.

So the point I wanted to make is that this listener's comment about liability is very interesting. We're constantly hearing about site breaches, with Troy Hunt's "Have I been Pwned" service constantly collecting massive troves of user login credentials. **All of that disappears** if a website no longer offers to store its users' password hashes. Why bother? It's just a potential nightmare. It's a liability. A secret they are not good at keeping. With 2/3rds of users today still not bothering to use a password manager, they are likely using crappy passwords. This renders the site vulnerable to attack, as 23andMe was, through no fault of theirs, because their users cannot be bothered to secure themselves.

So these attacks go unnoticed and unobserved because they can be conducted by autonomous distributed networks of bots. But that cannot happen if the user's eMail is dynamically included in the login process. This completely changes the attack dynamics. And don't get me wrong, I'm not suggesting that I think this is a good idea (except that it kinda is) because it definitely represents a nightmare for the other 1/3rd of us who are using password managers stuffed with long passwords we could never begin to memorize of even correctly enter into a password field.

The problem for our future is that eMail loop login makes a horrible kind of sense for any website that gets to choose whether or not they want the liability of storing the hashes of their users'

potentially crappy passwords. It's easy to imagine websites deciding that they'd prefer to ask their users to obtain a per-login token from their own eMail.

Mark Jones / @mjphd

Another in the "it was nice while it lasted" category. I have run into two sites that won't accept @duck.com emails. A previous solution, using a + in a gmail address, also has been thwarted. That allowed me to filter easily and reject sites of no interest or that got spammy. One service stripped the + and what was after out. The other trend you haven't mentioned is the insistence on a cell number. Those are unique, difficult to share and are harder to make throwaway. I was trying to set up services for a community organization only to have a couple of vendors balk because my cell was already associated with a different active account. Love the podcast, look forward to it every week and am happy it will continue.
Tested 6.1 and am loving that too.

As Mark wrote "it was nice while it lasted." Unfortunately, while it is possible to hide our real eMail address behind an identity-protecting eMail forwarding service, it's not possible to hide the fact that we're using an identity-protecting eMail forwarding service. And the fact that a site is stripping the '+' plus sign and what follows from eMail addresses is nothing short of rude. It's not their business to alter someone's eMail address. That seems so wrong. But the only leverage website visitors have is to choose not to play.

Normally, I would suggest that such eMail address discrimination would not become widespread. But if websites are asking for their visitors' eMail as a replacement for 3rd-party tracking, with the incentive of being paid more for ads served to identifiable Internet users, then those advertising interests who are paying **definitely** do not to pay more for anyone using an anonymizing eMail forwarding service. So you can bet that a site's advertisers will be telling the websites that they won't pay for anonymized eMail. And then, in turn, those sites will be telling their visitors exactly what our listener Mark Jones was told: "Please provide your primary eMail address, not a forwarding service."

Rob Powell / @RobPowe70442395

Hi Steve following recent discussions on the podcast I thought other listeners might also appreciate the below, it's a tool to detect when chrome extensions change owners
https://github.com/classvsoftware/under-new-management

I got a kick of of the name of the tool that Rob pointed to. It's posted on GitHub under the account of "classvsoftware" with the name "under new management". A couple of weeks ago we were noting that the authors and maintainers of Android freeware which had accumulated large install bases over time and earned some implicit trust from their users have effectively been cashing out their installed bases to less than scrupulous buyers who then take advantage of that installed base. Since the same thing could happen with browser extensions whose special position in our browsers makes this an even more pressing problem, I was glad that Rob brought this up. I went over to the GitHub site to see what the authors of this extension had to say. They explain:

"Intermittently checks your installed extensions to see if the developer information listed on the Chrome Web Store has changed. If anything is different, the extension icon will display a red badge, alerting you to the change."

And as to why this is an issue, they write:

"Extension developers are constantly getting offers to buy their extensions. In nearly every case, the people buying these extensions want to rip off the existing users. The users of these extensions have no idea an installed extension has changed hands, and may now be compromised. Under New Management gives users notice of the change of ownership, giving them a chance to make an informed decision about the software they're using."

In that text the phrase "*constantly getting offers to buy their extensions*" was a link. So I wondered what the author of this "under new management" extension might be linking to, and holy crap! ... the link was to another GitHub page titled "*Temptations of an open-source browser extension developer.*" But before I describe what's there, let me backup a bit. The extension in question is called "Hover Zoom Plus" and its author explains:

[HoverZoom] Zooms images & videos on all your favorite websites (Facebook, Amazon, etc). Hover your mouse over any image on the supported websites and the extension will automatically enlarge the image to its full size, making sure that it still fits into the browser window.

This is an open-source version of the original HoverZoom extension, which is now ***overrun*** *by malware and deleted from the store. In this version, all spyware has been removed, many bugs were fixed and new features were added. It doesn't collect any statistics whatsoever. The only permission it needs is to access data on all websites (to extract full images), and optional permissions to access browser history, download/save images, or get tab URLs for per-site configuration.*

This extension will never be sold out, and it will never compromise users' privacy. As a proof, please see the list of all takeover offers I have received over the last few years.

And this brings us back to the "*Temptations of an open-source browser extension developer*" page. It's so astonishing that I've made it the GRC shortcut of the week. If you're at all curious, visit https://grc.sc/965. The author starts off explaining:

Over the years, I have received many proposals to monetize this extension so I think I'll just start posting them here for fun (but not for profit). The main reason I continue to maintain this extension is because I can hardly trust others to not fall for one of these offers. I'm fortunate to have a job that pays well enough to allow me to keep my moral compass and ignore all of these propositions. I realize that not everyone has the same financial security.
So hopefully this thread will shed some light on what kind of pressure is put on extension developers.

And what follows really serves to put this into perspective. The first offer he posts is dated September 28th, 2015. It reads:

> *Hope this message finds you well. I'm a Strategic Partnerships Manager at a monetization platform for browser extensions. I am contacting you since I came across the extension 'Hover Zoom+' at the Chrome Store; I consider your product can help you bring profit by means of collaborating together. I would like to suggest a potential partnership between our companies that will significantly increase your revenues. Are you interested in discussing our offer in more details? Hope for positive feedback and ongoing cooperation.*

The most recent offer the author posted four weeks ago on February 14th, 2024, read:

> *I trust you're doing well! Currently, I'm exploring opportunities to grow my business by investing in Chrome extensions. Your extension Hover Zoom+ has caught my attention, and I am genuinely interested in discussing the possibility of acquiring it. We can discuss the price and complete the transaction securely through a reputable escrow services (escrow.com or cryptoexchange.com). Google supports the smooth transfer of extension ownership from one account to another, ensuring your gmail account remains unaffected. If you have any inquiries or if this aligns with your plans, feel free to reach out to us via this email. Looking forward to hearing from you!*

In between that first and last offer are more than – I counted them – 165 other offers this guy has received from parties interested in taking "HoverZoom+" off his hands. (I say "more than" because for a while I was not counting follow-ups as a separate offer. But there were so many of them that after a while I decided a more accurate count should include them.)

The danger to the users of extensions that are sold to unscrupulous buyers is that an originally benign extension might be altered to begin harvesting its user's browsing data as the US Federal Trade Commission has formally accused the Avast browser extension of doing. Since the purchasers of popular extensions with a large installed base have no interest in ongoing support and simply wish to maximize their return on investment, they will have a plan for somehow monetizing the extension's user base.

Given that, the value of the "under new management" extension becomes clear. So thanks for bringing it to our attention, Rob!

Max Feinleib / @MaxFeinleib

> *I'm sure I'm not the first person who's mentioned this, but SN 905 will be hard to beat for the shortest title: "1".*

Ha! I made the comment to Leo last week that "PQ3" might be the shortest podcast title we've ever had. Max was actually the second of two sharp listeners who said what amounted to "not so fast there, Gibson... what about podcast #905 that was titled with the single digit '1'?" That frighteningly low number was a reference to the password hashing iteration count some unfortunate LastPass users found when they went looking. So, thanks for catching that!

Passkeys vs 2FA

As I mentioned last week, I intended to title this week's podcast "Morris II" and talk about the creation of the first Generalized AI Internet worm. But while I was collecting and assembling this week's listener feedback, and started to answer one of our listener's very good questions, I decided that the question and its answer was very important and deserved everyone's full attention. So it was promoted to the primary focus of today's podcast, and assuming that nothing again preempts our discussion of "Morris II", we'll cover that next week.

Okay. So here's the Tweet that Stepthan Janssen sent:

Stephan / @stepjanssen

> *Hi Steve, I'm an avid SN listener and in part thanks to your information, moved from Lastpass to Bitwarden. Now that Passkeys are becoming more prevalent I'm noticing a Bitwarden popup when I try to enroll my Yubikey as a 2FA token. I'm tempted to give Passkeys in Bitwarden a try (in part due to your enthusiasm about it) but I'm also hesitant, since it feels like I'm losing and giving up my second factor. With Google, for example, using a Passkey allows one to login without providing anything else, which makes it feel like I no longer have MFA for my account.*
>
> *Am I right in thinking that using Passkeys will reduce my security if I'm now using a random password with a second factor, or am I missing something in my thinking?*

I'll give everyone the short version TL;DR first. But then, because as they say "it's complicated" I'll expand upon it. So here it is: In a properly operating world, Passkeys, all by its lonesome, provides far more security than any super-strong password plus any second factor. Period.

So let's start working our way through what has become a confusing authentication minefield.

First off: "Why does anyone use a second factor?" and I'm going to pointedly ignore the primary reason we, who listen to this podcast, use second login factors, which is mostly because they're cool. It feels a little secret-agent-like to have to go lookup that time varying code to complete an authentication. It definitely provides the impression of having much greater security even if there are still ways around it. And we've talked about the ways around it. Multi-factor authentication can be and is actively compromised when there's some strong need to do so. There are two ways this can be accomplished:

The first is arranging to insert a man in the middle. This is done in today's world by arranging to trick an unwitting user into going to a fake login page. They provide their username and password, and are then prompted for the second factor. They provide that to the man-in-the-middle, who turns around and immediately logs in as them before that ephemeral 6-digit code expires. This is not easily done because it requires a higher-end attacker and some setup. But it **is** being done. Normally this only intercepts a user's username and password which does give the bad guys more permanent account access until the password is changed. But when you think about it, once you have that MITM proxy in place to intercept the username and password, the addition of an ephemeral second factor adds very little to the user's actual security.

The second way the MFA system can be defeated is the same way any site breach, with a loss of their authentication database, can defeat the security for their users: Just as the authentication database contains a hashed password which is used to verify the user's freshly- provided login attempt password, that authentication database must also contain the shared MFA secret which keys the user's TOTP – time-based one time password. So, if that MFA secret can be stolen along with the username and password hash, then bad guys have potentially acquired all of the secrets they need to impersonate the site's users anytime they wish.

The key here is my use of the term "secrets", because today's time-based one time passwords are based upon shared secrets. The site knows the secret key and your authenticator app has the same secret key. That's what allows the authenticator to generate a 6-digit code based upon the time of day, and for the server, knowing the same time of day, to generate the same code and compare them.

And "storing secrets" is the crucial difference in the Passkeys system. This was also true with SQRL's technology which I articulated on this podcast more than 10 years ago. What I used to say was that "SQRL gives websites no secrets to keep" – and exactly the same is true of Passkeys.

The crucial difference is secret versus public key technology. That's what makes Passkeys so very different from everything that came before it (except of course SQRL which works similarly). With Passkeys, the user's private key never leaves the authentication device end and the website only ever receives the user's matching public key. And the only thing that the website's public key can be used for is verifying the signature of a unique challenge:

The website sends the user a never-before-used large random number to sign. Since the large random number has never been used before, its signature will also never have been used before, so no form of replay attack is possible. The user's Passkey agent signs the challenge and sends it back to the website, signed. The website then uses its matching public key to verify the user agent's Passkey signature. And, crucially, verifying signatures is all it can do with that public key. So if that key were to be stolen, no one cares.

The reason I felt that Stephan's question deserved our full attention is that none of this crucial difference between traditional username and password – and even multi-factor authentication – versus Passkeys is in any way apparent to users. Users just see yet another thing – another way to log on. What's impossible to see and to appreciate from the user-facing side is how completely and importantly different Passkeys is from everything that has come before.

So, to respond to Stephan's uneasiness about whether he's losing anything, sacrificing anything, or giving up anything by choosing to use a Passkey instead of having traditional login in – even when it's augmented with multi-factor authentication, the bright flashing neon answer is "No!"

The best way to think about multi-factor authentication is that it was a last-ditch temporary Band-Aid that was added as an emergency measure in an attempt to do something to shore up the existing ecosystem of fundamentally weak and creaky server-side secret-based authentication. Everything about Passkeys is superior to everything that has come before it. Well, except for SQRL, but that argument is now academic.

However, while everything I just said is true in theory it may not be completely true in practice: As we know, security is always about the weakest link in the chain; and that's every bit as true for authentication. If a Passkey is setup for website login, that system of authentication will be the most bulletproof solution mankind knows how to design today. But if that website also still supports username and password authentication, with or without any second factors, then that username and password fallback will reduce the entire system's effective security.

The problem is that while Passkeys are actually vastly more secure than username and password login, they're primarily seen as being more convenient. In other words, it's not necessarily the case that enabling the use of a Passkey will simultaneously and robustly disable all use of any preceding username and password. And that's what we really want.

If you've never been to a website which asks you to create an account for the first time ever, and offers to let you use a Passkey, you're golden, since you won't have ever given that website any secrets to keep. It cannot disclose what it doesn't know. The question is, will websites that start allowing the use of Passkeys allow their users to then erase all traces of, or firmly disable the use of, their previous username and password for login? Because if they don't, the use of Passkeys is only giving them more convenience, but not also the absolute security it otherwise could if it were to be the singular login option – aside from the ever-present eMail loop, of course.

This problem was something that my design of SQRL anticipated. After its user had become comfortable with the way SQRL operated, and had printed out the necessary identity backups, they were able to turn on a setting that requested every site they subsequently visited to please disable any and all alternative means of logging in – including specifically the eMail loop fallback. Either we have true maximum security, or we don't.

So here are our takeaways from all this:

The crypto technology which is well hidden inside Passkeys totally blows away both username and password and even MFA authentication, which all rely upon server-side secrets being kept. That's their universal weakness which is what makes Passkeys not only more convenient but also far more secure. So whenever given the choice, set up a Passkey without a second thought and without looking back.

Secondly, look around to see whether there's any way to disable any other previous less secure login fallback. If you cannot disable username and password login, but strengthening it with MFA is possible, by all means do so. Adding MFA is the best way to make username and password login safer against attack, even if you're not still using your password and have switched over to Passkeys.

One thing you **CAN** definitely do is remove your old username and password from your password manager. And if for some reason you cannot remove it, change it to something bogus, like a long string of pound (#) signs. That will be your sign that this account uses Passkeys. And by removing your password manager's storage of the account's true password you're protected from any compromise of that password manager or its cloud backup provider.

The big takeaway here is that even though they may not look all that different from the outside, the technology underlying Passkeys makes them an unreserved win over everything that has come before.

Now that our password managers are fully supporting passkeys natively, which is what we've been waiting for, whenever possible, make the switch and then follow-up by doing anything you can to remove the use of the username and password login that preceded it.

And Stephan, thank you for the question. This has obviously been an issue that's needed some time and clarification. :)