▲ A Yubico FAQ about passkeys (yubico.com)

121 points by Shank on Dec 10, 2022 | hide | past | favorite | 131 comments

▲ account-5 on Dec 10, 2022 | next [–]

I will probably use passwords until I am forcibly unable to. I'm happy with my password manager.

Tying login credentials to a hardware device that can easily be lost or stolen, and that (at least from my understanding) are hard to transfer to new devices seems like a risk I'm unwilling to take. And a step back.

I'm not adverse to replacing passwords with public/private key cryptography, but I need to be in control of it (not some multinational org) and I don't want it tied to a single hardware device.

▲ goodoldneon on Dec 10, 2022 | parent | next [–]

I have 2 yubikeys and always register them both. So if one is lost or broken then I still have another to log into sites.

As for theft, if you're using your yubikey as a 2nd factor then they still can't log in because they don't know your password. If you're using your yubikey as both factors then it either has a PIN or biometrics, so the thief can't log in.

My main use for my yubikey is for SSH. I have PIN-protected SSH with a private key that can't be exfiltrated

▲ Robadob on Dec 10, 2022 | root | parent | next [–]

I have two yubikeys too, but I definitely found services which didn't allow me to register more than one when I first set them up (a couple of years ago now).

▲ Semaphor on Dec 10, 2022 | root | parent | next [–]

Not sure how it used to be, but the only braindead implementation like that I've seen was AWS.

▲ arianvanp on Dec 10, 2022 | root | parent | next [–]

They fixed this a few weeks ago.

▲ chrishynes on Dec 10, 2022 | root | parent | next [–]

They announced a fix but it hasn't rolled out everywhere yet -- it requires converting to a new "Amazon Web Service" login that is separate from your normal amazon.com account

▲ bostik on Dec 10, 2022 | root | parent | prev | next [–]

Oh cool, this has been my number one complaint about AWS credentials management since early 2016 or so. Over the years I've talked to a number of their project and program managers and until 2019 (that is, before the plague) they all told me that they were aware of the problem but had no plans - or even *intent* to fix it. Apparently the one-set-of-credentials constraint is so deeply tied into their platform that fixing it was simply deemed infeasible.

Appears that while they couldn't fix it, they have gone ahead with a workaround and made multi-device 2FA support a feature in a potential auth replacement system.

Must have been a particularly frequent and loud complaint.

▲ tuatoru on Dec 10, 2022 | root | parent | prev | next [–]

I bought four yubikeys back in 2012, two pairs as recommended - one in use, one backup. At the time Yubi had a video up of their keys being run over and otherwise maltreated, and still working.

One of my keys failed spontaneously after a just few weeks in my pocket, on a ring with my other keys.

I decided to stop using yubikeys. A password manager has worked very well for me ever since.

▲ ViViDboarder on Dec 10, 2022 | root | parent | next [–]

I've had one on my keychain for at least 5 years now with no issues. It would have been longer but I did switch to a USB-C key two years ago. I kept the old one enrolled as my backup key.

I use a password manager, but my Yubikey has all my 2FA. Including TOTP.

▲ Jayschwa on Dec 10, 2022 | root | parent | prev | next [–]

I had a YubiKey fail when I touched it and a static shock wiped its memory. I was able to reinitialize it using their app, but the previously loaded key was lost.

▲ spiffytech on Dec 10, 2022 | root | parent | prev | next [–]

> I have 2 yubikeys and always register them both

How does this work? Do you have to physically retrieve both any time you sign up with a new service?

If that's how it works, it sounds like a choice between two bad options: either having both of them handy at the same time, hurting redundancy, or registering them at separate times, hoping you remember to do both. Is that the case?

▲ goodoldneon on Dec 10, 2022 | root | parent | next [–]

That's what I do. I keep my backup in a fireproof, airtight container

▲ jdfhfjdhjdhdfj on Dec 11, 2022 | root | parent | next [–]

excuse me if i think this is BS.

You will have to retrieve your key when you sign to a new service. at this point you risk losing both.

or are you saying you cloned the private keys and know how to correctly advance the counters when you need the backup?

▲ goodoldneon on Dec 11, 2022 | root | parent | next [–]

I don't use 2 factor for most sites. Do I really care if my HN account is compromised? So for sites that I do care about (e.g. GitHub), yes I register both. It isn't as inconvenient as you'd think because I don't sign up for many sites.

Since I use my Yubikey for SSH, I'm frequently adding both Yubikeys as authorized keys on machines. But I keep an authorized_keys file with both public keys so I don't need to physically grab my backup

▲ illiac786 on Dec 11, 2022 | root | parent | prev | next [–]

But if you register both every time, it means you always carry both yubikeys with you, right? I hope you don't have both on the same keyring, but if you carry both around, the resilience/redundancy of that setup is lacking in my view.

▲ orev on Dec 10, 2022 | root | parent | prev | next [–]

The elephant in the room is that this just isn't feasible for "normal" (non-IT) people, and those people are usually the ones who need strong protections, as they're the ones with bad habits like choosing bad passwords, reusing passwords, forgetting them, etc.

▲ ilyt on Dec 10, 2022 | root | parent | next [–]

I'd think password managers would be fine here. Also, don't fuck up other user's experience just because average one is clueless...

▲ goodoldneon on Dec 10, 2022 | root | parent | next [–]

I use a password manager too. The yubikey is usually my second factor

▲ goodoldneon on Dec 10, 2022 | root | parent | prev | next [–]

You’re totally right. I never recommend Yubikeys to non-technical people

▲ neatze on Dec 10, 2022 | root | parent | prev | next [–]

sudo, screen lock, luks, ssh, pgp, weblogins, keepass are all setup with yubikeys. I wish I could generate my own private fido key in similar way as pgp.

▲ vladvasiliu on Dec 10, 2022 | root | parent | next [–]

Yes, I think it would be great to be able to create the key otuside of the physical hardware token, just like you can do with the GPG applet on the Yubkey. Store the original on some offline, encrypted media, ideally in multiple places.

This way, you don't need to have two yubikeys on hand and register them both on every new site. But in case you lose, it's not impossible to create a new one. Yet, obtaining the key from the hardware token shouldn't be any more possible than with the current system.

▲ Godel_unicode on Dec 10, 2022 | root | parent | next [–]

These two goals are mutually exclusive. If you let people generate their own key material outside the enclave, people will be emailing it to themselves just in case.

▲ ilyt on Dec 10, 2022 | root | parent | next [–]

Well, non-stupid people also use yubikeys, so they are not mutually exclusive

▲ Godel_unicode on Dec 11, 2022 | root | parent | next [–]

Maybe, but there are also a lot of people who don’t understand the point of yubikeys that want to use them but export the secrets. Those people cannot be trusted with their own security and need to be protected from themselves.

▲ neatze on Dec 14, 2022 | root | parent | next [–]

No one mentioned exporting keys, but importing to yubikey (one way) same as it is already done with gpg on air gaped machine versus generating fido private key on reset within yubkey itself.

▲ Godel_unicode on Dec 15, 2022 | root | parent | next [–]

1) yes they did, I suggest you look again. 2) That’s the same thing. The end state is wanting the key material outside of the secure element, which is foolish.

▲ ilyt on Dec 11, 2022 | root | parent | prev | next [–]

Nope, let them be burned, the sooner the better, before they own something actually valuable behind the account

▲ Godel_unicode on Dec 11, 2022 | root | parent | next [–]

What a truly psychopathic take, I really hope you’re not making decisions that effect people.

▲ diggernet on Dec 12, 2022 | root | parent | next [–]

I have the same opinion of people who want to control others because they "need to be protected from themselves".

▲ Godel_unicode on Dec 15, 2022 | root | parent | next [–]

So I take it you’re against food safety standards, seatbelt laws, taxes on cigarettes, the clean air act, payday loan reforms, etc.

Most people will never understand this subject, and the fact that they don’t need to is a good thing. If you want a hobby, great, you can go figure out how to do that. The default needs to be safety.

▲ account-5 on Dec 10, 2022 | root | parent | prev | next [–]

This is what I would require before migrating. Opensource local only Fido key generation.

But like I said in the comment below, I suspect these vendors are only here for vendor lock-in and targeted advertising. And the websites enabling webauthn are likely to require you to use a vendors app/service. Like banking sites today not allowing me to utilise keepassxc totp functionality trying to force me to download their app or SMS.

▲ cmdli on Dec 10, 2022 | root | parent | prev | next [–]

Software-based passkeys can do this! Bulwark Passkey (https://bulwark.id) is a virtual Yubikey-like device, and you can back up the credentials file however you like.

▲ sebk on Dec 10, 2022 | parent | prev | next [–]

This concern is valid depending on your threat model, but improvements are quickly coming to this area. Apple, Microsoft, and Google have announced that they will allow syncing credentials between devices in their ecosystem (fully end-to-end encrypted, like a password manager), and several password managers have started to add software implementation of FIDO authenticators to their offerings. Hopefully in the near future we'll see the big vendors open up their APIs so that password managers can use the device hardware as well.

Of course, multi-device passkeys will not work for high assurance situations, as TFA calls out, but that's not likely the case for any of the services you currently use a password manager for anyway.

As it stands today, using passkeys with a software implementation, barring implementation bugs, is no less safe from credential loss than passwords in a password manager.

▲ bbarnett on Dec 10, 2022 | root | parent | next [–]

Apple, Microsoft, and Google have announced that they will allow syncing credentials

While your comment is fair, and I know you were just providing info, from where I sit, these are the sort of entities I never, ever involve in any form of private work, security, or communication.

▲ toomuchtodo on Dec 10, 2022 | root | parent | next [–]

This is a minority opinion though when compared to usage and market share of these orgs.

▲ mistrial9 on Dec 10, 2022 | root | parent | prev | next [–]

as a US citizen, I absolutely do not accept these three giant companies running my access to financial services and governmental functions; just, no. Secondly, I will say that the entire surveillance-capitalism ideas of constant-on ID comes from a model of money lending that was built in the 1960s, ultimately by VISA and MasterCard. This worked well enough to generate the holders massive capital. no, no, no

▲ PhilippGille on Dec 10, 2022 | parent | prev | next [–]

> Tying login credentials to a hardware device

The article specifically states (about passkeys as opposed to Yubikeys):

> They're different because Platform created passkeys will be copyable by default using the credentials for the underlying cloud account (plus maybe an additional password manager sync passphrase), whereas passkeys in YubiKeys are bound to the YubiKey's physical hardware where they can't be copied.

That brings its own challenges of course (e.g. you have to trust the Cloud syncing and E2EE by the vendors).

▲ account-5 on Dec 10, 2022 | root | parent | next [–]

Apologies I was trying to make my second condition about these providers. I don't want hardware specific credentials, or <insert multinational cloud provider> controlled credentials.

The first introduces risks with damaged or stolen hardware, not to mention migration.

The second introduces lack of control and enables these companies to track you. It's not lost on me that nearly all the companies offering authenticator apps are basically advertising companies: Google, Microsoft, Apple. They're getting in at the start for a reason

▲ georgyo on Dec 10, 2022 | root | parent | next [–]

It does not need to be hardware specific. After all, talking to hardware is just software.

Your password manger could implement passkey support. And I suspect that 1password will do that soon with their recent acquisition.

The real benefit to passkeys is that password isn't sent over the wire, only the proof that you have the secret. Similar to SSH keys. This prevents a whole range of attacks.

▲ aniforprez on Dec 10, 2022 | root | parent | next [–]

1Password already has passkey support in development. They have a demo[0] which works on the latest version of the browser extension. It even syncs with my Macbook and Windows machines though unfortunately doesn't work on Android yet

[0] https://www.future.1password.com/passkeys/#demo

▲ account-5 on Dec 10, 2022 | root | parent | prev | next [–]

I use keepassxc, I would love to manage my private/public keys in the same way I manage my passwords. I'm in complete control.

But from my reading, and I believe the way this will be implemented but services, I doubt very much you'll be about to do this. Certainly at the moment it looks like vendor lock-in to me. Vendor lock-in with a heavy dose of targeted advertising.

▲ brookst on Dec 10, 2022 | root | parent | prev | next [–]

I’m not seeing the threat model. Assuming E2EE of the password manager, how does it track you when syncing keys across devices?

Also, how is Microsoft not an enterprise security company? Their revenue from security products is probably 100x that from advertising.

▲ account-5 on Dec 10, 2022 | root | parent | next [–]

Apologies I don't understand your first paragraph. I'm preferring my password manager because it doesn't track me. It's offline and local.

Microsoft might be an enterprise security company. But they are clearly all about tracking and targeting their users too. The two things are not mutually exclusive.

▲ brookst on Dec 10, 2022 | root | parent | next [–]

Re: first point, I don’t see how E2EZe sync of keychains exposes you to targeted advertising. My understanding of the systems is that they use a single encrypted DB that does not expose either the sites, usernames, or passwords to the provider. I’d welcome correction if you know otherwise.

Re: Microsoft, I suppose there’s no harm in being maximally suspicious. But I have some knowledge of their systems and Google’s and Microsoft does not have , for instance, a single user profile that joins interests and infers demographic data like Google does. I expect the same is true for Apple.

▲ account-5 on Dec 10, 2022 | root | parent | next [–]

My concern is that of relying on one of these vendors for managing the passkeys. And their knowledge of which sites I have accounts with. And my suspicion you'll have an extremely hard time migrating devices and vendors.

I've no internal knowledge of MS systems but the fact that they make it hard to use their operating system without an online account and you can't opt out of all telemetry and the fact it's default opt-in for all telemetry leads me to be extremely suspicious of them. I'm not saying they're any worse than Google or Apple.

▲ brookst on Dec 10, 2022 | root | parent | next [–]

I’m confused about know they know which sites you have passkeys for. To the best of my knowledge, neither Apple’s nor Google’s password managers expose that to those companies.

See for instance: https://support.apple.com/guide/security/secure-keychain-syn...

▲ account-5 on Dec 10, 2022 | root | parent | next [–]

I have a feeling we might be talking about different things. I'm not talking about password managers, and having never used any password manager that wasn't opensource and offline I'm in no place to comment. What I do know is that at no point do I want these companies in charge or gatekeeping the keys (so to speak) to my kingdom. Their motives aren't ultruistic and are only self serving. In the case of Google certainly you could end up locked out of all your accounts if their AI decides to shut your account down without recourse or a Twitter/HN thread about it.

▲ brookst on Dec 10, 2022 | root | parent | next [–]

I think we’re talking about the same thing. There is no gatekeeping of keys. There IS gatekeeping of the services that syncs keys across devices.

If one of these companies locks you out of your account for any reason, you don’t lose access to the keys. They are not removed from your devices. You do lose the service that moves them between devices.

Anyway, I appreciate you sharing your perspective and I admire your commitment to only use fully open services. Just saying the closed ones may not be as malicious or a usable as you think.

▲ cmdli on Dec 10, 2022 | parent | prev | next [–]

I 100% agree. I really like the idea of getting rid of passwords, but I wanted to be able to control my own information. That is why I built Bulwark Passkey (https://bulwark.id), an open-source virtual passkey manager which allows you to export your credentials out of it, or sync them across devices.

Passkeys have a lot of advantages, but the current plans for implementation are rather limited. If you're interested in passkeys that you can actually control, please check Bulwark Passkey out!

▲ diggernet on Dec 12, 2022 | root | parent | next [–]

It looks like the only options are local-only (single device) or cloud-synced (bulwark cloud). I want credentials that are not tied to a single device OR tied to a cloud service. I want to be able to make backups to be used in the case of device failure or loss, and I don't want to be tied to some "underlying cloud account" (in Yubico's words). Just standalone, backupable credentials. Can you do that?

▲ worldsavior on Dec 10, 2022 | parent | prev | next [–]

Using passwords is not something very comfortable, and needing to remember a password to a password manager, has some risks, like suddenly forgetting. Using a yubikey is also not very practical since it can be stolen or lost. A phone is always with us, and has small chances of being lost (also there are backups).

▲ bennysonething on Dec 10, 2022 | root | parent | next [–]

I use a yubikey to secure my passwordsafe file https://pwsafe.org/

Password safe runs on multiple operating systems. I sync my file to cloud storage so it's up to date on all my devices

I have a backup key in a drawer and one on me at all times.

I really need to buy a few more backup keys.

It's not a perfect set up but it works for me.

▲ waych on Dec 10, 2022 | root | parent | next [–]

Consider trying SyncThing to sync your files between your personal devices, rather than entrusting access to your data to a cloud provider.

I did this recently and the experience has been surprisingly way easier and better than I could have imagined.

▲ bennysonething on Dec 10, 2022 | root | parent | next [–]

That looks great. Looks it might be viable for me. Security risk in my setup is is cloud storage. And also I'm relying on a password safe Dropbox sync app.

▲ fbdab103 on Dec 10, 2022 | root | parent | prev | next [–]

Phone has a significantly higher chance of breaking.

Owner of a Pixel that just died in my hands. Had I not been with company that day, I would have been in a real situation.

Edit: also, if I keep it on the phone, will Apple/Google deign grant me access to export my secrets on demand or will they deem I cannot be trusted with such power?

▲ worldsavior on Dec 10, 2022 | root | parent | next [–]

> Phone has a significantly higher chance of breaking.

Breaking in is very hard since the security model for phones is very high, especially in cases of Pixels and iPhones. Even if a pin is used, the security chip makes it very hard to break in. If you mean also in case of losing it: Not as a small device that has much more higher chances of being lost.

> Edit: also, if I keep it on the phone, will Apple/Google deign grant me access to export my secrets on demand or will they deem I cannot be trusted with such power?

Don't know about that, since passkeys are still pretty new, but probably it will be an option. Does a yubikey allows you to export secrets?

Edit: typo

▲ Godel_unicode on Dec 10, 2022 | root | parent | next [–]

> Does a yubikey allows you to export secrets

No, that would defeat the purpose. If you want exportable secrets just use a thumb drive.

▲ worldsavior on Dec 10, 2022 | root | parent | next [–]

That was a rhetorical question.

▲ account-5 on Dec 10, 2022 | root | parent | prev | next [–]

A phone has the same chances of being lost and I would say a higher chance of being stolen.

My question would be who is in control of the backups? The same vendors who want to lock you into their ecosystem?

▲ bbarnett on Dec 10, 2022 | root | parent | next [–]

And the same ones that also lock you out without recourse?

▲ worldsavior on Dec 10, 2022 | root | parent | prev | next [–]

> A phone has the same chances of being lost and I would say a higher chance of being stolen.

Not the same as a small key that can be stolen.

> My question would be who is in control of the backups? The same vendors who want to lock you into their ecosystem?

You, as it's encrypted (in case of apple, not sure about Google). Backup is up to you, you can also backup to other services such as nextcloud.

▲ sudhirj on Dec 10, 2022 | parent | prev | next [–]

Why is a passkey any different? Can’t one think of it as a password stored in a password manager? Is it more that you don’t want your browser to function as the password manager?

There seems to be a API already for a browser to delegate passkey storage to a Yubikey, can’t a password manager do the same thing?

▲ cmdli on Dec 10, 2022 | root | parent | next [–]

Right now APIs for third parties are fairly limited, so it can be difficult to have support for passkeys with password managers. I was able to solve it for Bulwark Passkey (https://bulwark.id) by emulating a USB device and building the WebAuthN protocol off of that. This means that it supports all browsers, but also means that iOS support will be more difficult, though not impossible.

▲ ilyt on Dec 10, 2022 | root | parent | prev | next [–]

Browsers having just API for password managers is separate discussion but yes, that should happen.

Password manager's job should not be to scrape pages for login fields. That still could be a separate plugin (if browser would not do job good enough), but password manager should really get credentials name, some metadata, notes and password to store, nothing more

▲ izacus on Dec 10, 2022 | parent | prev | next [–]

> I will probably use passwords until I am forcibly unable to. I'm happy with my password manager.

A random password stored in a password manager is literally what a passkey is. Several password managers have announced support for storing passkeys just like your passwords.

▲ Renevith on Dec 10, 2022 | root | parent | next [–]

"A random password stored in a password manager is literally what a passkey is."

This is not quite true; a passkey is still somewhat more secure than a password in this scenario.

With passwords, the password itself is sent to the website and a hashed version is stored in their database. If someone can intercept your connection to the site (MitM) or can access their database (hacker), they might be able to get your password and log in as you, especially if the website doesn't implement their hashing very well.

These attack vectors don't apply to passkeys, even generated and synced by a software password manager: the private key never leaves your computer, only the corresponding pubic key. The website generates a unique challenge on each login that can only be solved by your private key, so intercepting a particular challenge or stealing the public key from the website database doesn't let anyone log in as you later.

▲ arcanemachiner on Dec 10, 2022 | root | parent | prev | next [–]

I've been keeping track of this GitHub issue to follow the progress of getting the feature in FOSS password managers:

https://github.com/keepassxreboot/keepassxc/issues/8214

It appears that we are still in the "walled garden bullshit" phase, so I'll pass for now.

▲ Biganon on Dec 10, 2022 | root | parent | prev | next [–]

I don't understand your last sentence. If a random password stored in a password manager is a passkey, then all providers already support them

▲ jjnoakes on Dec 10, 2022 | root | parent | next [–]

Passkey is about the random password stored in the password manager AND the protocol between the browser and the password manager to ensure the password isn't phished. It is the protocol that needs implementation.

▲ jolmg on Dec 10, 2022 | parent | prev | next [–]

> and that (at least from my understanding) are hard to transfer to new devices seems like a risk I'm unwilling to take.

It's not really hard. They're quite flexible. You can't read the key from them, but you can write to them. You can generate the key outside, write it, save the backup somewhere you trust to be safe according to your threat model, and that's it. If you lose the device, buy a new one, write the key from the backup. The backup can also be used directly if you can't buy a new device for whatever reason.

At least, this is how it works when used as an OpenPGP smartcard.

▲ AugustoCAS on Dec 10, 2022 | parent | prev | next [–]

I use keys as my default way of login but I have 'phishable' backup methods (OTP codes or backup codes).

For me, the main thing is that using the keys by default make my accounts more secure and I would only use the phishable methods in case of emergency (if my house goes up in flames with my keys, computer, mobile, etc). One important thing for me was to remove my mobile number for SMS OTP or password reset, as sim cloning is a posibility.

And a password manager is still indispensable as most sites don't accept hardware keys at all.

▲ ilyt on Dec 10, 2022 | parent | prev | next [–]

With how "secure" average app is I think there is probably much higher chance of them just fucking up than someone that uses password manager and doesn't run random shit off internet getting hacked.

If it was "here is your private key, you can use it in sofware or you can load it on a key and just use" I'd be much calmer about it but the whole push "your authentication and sometimes whole identity is inseparably tied to your device" is frankly scary and disgusting.

▲ JustSomeNobody on Dec 12, 2022 | parent | prev | next [–]

I have one of the old blue keys. Never found a good use for it. I don't have my keys with me at my desk, so I never put it on my keychain. So then it's just laying around and gets shuffled of the desk and into a drawer.

Meanwhile, my phone is always near and bitwarden just works.

▲ AeroNotix on Dec 10, 2022 | parent | prev | next [–]

You don't have to be tied to a single device. A common set up is a root yubikey you create children of. The root key is held in a safe location and the children have either copies or even better, short live keys signed by the root.

▲ sebk on Dec 10, 2022 | root | parent | next [–]

YubiKeys can't have copies of themselves, that's a big portion of their selling point. As far as I know and strictily in FIDO, there is no solution here. The closest that Yubico has is this draft: https://github.com/Yubico/webauthn-recovery-extension which will roughly make an authenticator register two keys with an RP. The draft itself is not implemented anywhere as far as I know and while better than the current state of hardware key backups, it's still not problem free.

I personally would *love* to see it implemented, so it can be used for logging in to the service that provides a WebAuthn sync fabric.

▲ waych on Dec 10, 2022 | root | parent | prev | next [–]

I don't know about this "root key" setup you describe, but you can certainly program sets of yubikeys with the same OTP secrets at setup time, resulting in cloned/backup keys. The OTP secrets can also be stored (presumably offline) to create new copies of the key in case of recovery.

▲ sigzero on Dec 10, 2022 | root | parent | prev | next [–]

"passkeys in YubiKeys are bound to the YubiKey’s physical hardware where they can’t be copied."

Unless you are talking about something entirely different that scenario isn't possible.

▲ account-5 on Dec 10, 2022 | root | parent | prev | next [–]

I'll have to take your word for it. That all seems complicated to me, maybe it's not and is just out of my frame of reference.

▲ xoa on Dec 10, 2022 | prev | next [–]

It's great to see ending the insanity of symmetric authentication (with passwords) is finally, finally gathering momentum. It's been frustrating because there were various potential efforts going back to the 00s at least, I remember logging into some sites with client certs. Somehow though the right combination of UI and broad support never happened despite the endless ridiculousness of leaks due to inevitable server side compromise, and then ever more bandaids on top like adaptive hash functions and password managers. Really looking forward to the day of all that fading into an (admittedly very) long tail of legacy.

One place I'm kind of sorry hasn't been on the forefront of adoption though and gets into gear soon is web UIs, which often control some pretty important stuff. We run things like OPNsense for firewall/gateway needs, TrueNAS, various controllers, hardware that these days has a web UI etc. With "internal" applications it's already a lot more acceptable to slap heavier access requirements on them, only exposing them via VPN, but it'd be nice to be able to require hardware bound passkeys (to use Yubico's terminology) for web auth (recovery can be done via console if ever required).

▲ jtaft on Dec 10, 2022 | parent | next [–]

What’s your take on accessing systems accepting passkeys on a shared computer? Or if you lose access to your devices?

I think exporting passkeys could be useful.

I wonder what phishing attempts will look like.

▲ judge2020 on Dec 10, 2022 | root | parent | next [–]

On a shared computer, caBLE (“scan a QR code on another device”)[0] is the intended solution, since it’s a temporary way to authenticate and doesn’t save access to the target computer or anything.

0: https://i1.wp.com/9to5google.com/wp-content/uploads/sites/4/...

▲ xoa on Dec 10, 2022 | root | parent | prev | next [–]

>What’s your take on accessing systems accepting passkeys on a shared computer?

Plugging a token into a shared system is superior to passwords as well. Easy, somewhat safer if the system is compromised vs any given password (since an attacker would only be able to perform an online hot attack, and only on things the user was also doing assuming touch is required), much safer and easier compared to a password manager. Externalizing authentication on a shared system into something owned by the user is a pure win IMO for those who need to use them, and if anything is a particular win in that case compared to those who get to use only their own computers.

>Or if you lose access to your devices?

There is no one answer to this and I'm sure UX will evolve over time. As I said, in many cases there is a natural fallback in the form of local access console, same as if somehow all SSH keys or the like were lost. Ideally in general everyone would have multiple tokens, with at least one serving as a backup. Alternatively or in addition, good old backup codes (printed keys, either whole or n-of-m splits) on paper are a reasonable choice in some cases IMO. Some places that have a personal connection may simply have an IRL fallback, ie., your employer would just have you go to IT, or your bank might have you go into a branch in person with ID for a reset. In the future, there could be optional hardware solutions that allowed backing up exclusively to another key. I don't see the problem though as much different, than having hundreds/thousands of passwords to deal with.

>I wonder what phishing attempts will look like.

Pure remote "phishing" as it currently exists just isn't going to be possible with hardware based keys. But online hot attacks still will be, so I'd expect that will be that path taken (though it'll be harder). Stuff like getting people to run programs or waiting for them to access something sensitive on a rooted system such that they do a completely legitimate authentication flow but then additional actions are performed using it. But that's far more limited a threat surface than right now, and also will have more tractable technical counters. Still an improvement.

I guess for a while the other "phishing" that might linger will simply be trying to use the inevitable legacy workarounds that you yourself asked about. Backup codes could certainly be phished in principle, social engineering tech support that "so and so lost access to their devices!", etc. But I think that will fade in effectiveness as time goes on, so might as well get to it. At the end of the day, symmetric shared auth is just insanity and asymmetric is simply fundamentally superior.

▲ cmdli on Dec 10, 2022 | root | parent | prev | next [–]

If you want a passkey manager that allows you to export credentials, I would like to plug my own solution, Bulwark Passkey (https://bulwark.id). It's open-source and lets you export your passkeys outside of the system, as well as sync them across multiple devices, much like a password manager.

▲ ryanklee on Dec 10, 2022 | prev | next [–]

I've tried multiple times to work YubiKeys into my workflow and failed each time. The key needs to be with me *all the time* from room to room, building to building, location to location, not get lost, from change of clothes to change of clothes, and not end up in the laundry, and this is hard, hard, hard.

Who has managed to do this successfully and without losing their minds?

I hate passwords and would love to complement them with better security factors than my phone alone or find a way to move away from them entirely, but YubiKeys et al do not seem consumer viable to me. And I've tried.

▲ sebzim4500 on Dec 10, 2022 | parent | next [–]

I just leave it on my keyring.

Do you have a code to enter your building or something?

▲ ryanklee on Dec 10, 2022 | root | parent | next [–]

I don't need my YubiKey just at work. I need it with me all the time regardless of whether I am even wearing clothes.

▲ FeistySkink on Dec 10, 2022 | root | parent | next [–]

OK, I'll bite, where are you logging in naked? Some eyes wide shut situation?

▲ ryanklee on Dec 10, 2022 | root | parent | next [–]

Because in my day to day life wearing clothes is not a prerequisite for interacting with technology. I frequently exercise. This requires many showers and many changes of clothes. I walk around in the morning just wearing my boxers for the first hour of the day or the last hour of the night. This isn't exactly abnormal behavior. I always have my phone around but always having my phone plus always having my YubiKey is a step up in difficulty and not a linear one. Any successful auth solution better be prepared to solve such behavioral issues or they won't ever reach sufficient adoption levels.

▲ sho on Dec 11, 2022 | root | parent | next [–]

> This isn't exactly abnormal behavior

The exercise and showers maybe, but feeling like you need your yubikey with you literally 24 hours a day certainly is. I use them extensively and with everything properly set up perhaps need it a couple of times a week at best. Having it on a keychain or in a backpack is fine.

Hate to say "you're doing it wrong" but you (or your company) certainly seem to be doing something wrong to have this level of hassle. No-one would use them if they were this much trouble.

▲ ryanklee on Dec 13, 2022 | root | parent | next [–]

Certainly interested in hearing what I might be doing wrong.

▲ FeistySkink on Dec 11, 2022 | root | parent | prev | next [–]

You could put it on a chain around your neck. I don't know the specifics of your situation if that's feasible. Also how often do you have to use 2FA? I've been using hardware keys for a few years. I have one plugged into my dock and one in my backpack. Which seems to be enough. So far I haven't been caught off guard.

▲ sebzim4500 on Dec 10, 2022 | root | parent | prev | next [–]

Maybe this is a cultural difference but I can't imagine why I would need to sign into something while naked.

▲ ryanklee on Dec 10, 2022 | root | parent | next [–]

Please see my reply to sibling comment.

▲ lokar on Dec 10, 2022 | parent | prev | next [–]

I have one on my keychain, one in each computer and one in a safe

▲ brookst on Dec 10, 2022 | root | parent | next [–]

And you enroll them all for each service?

▲ stavros on Dec 10, 2022 | root | parent | next [–]

Yes. It's not super fun, but I only need to do it once per service.

I even built a Solo 2 key into my framework laptop:

https://imgz.org/i6Gn2fyS/

▲ vlmutolo on Dec 10, 2022 | root | parent | next [–]

That's incredible. I can't imagine the Framework people ever thought of this particular extension card use case.

I wonder if something like this could sell on their third-party extension card store.

▲ stavros on Dec 10, 2022 | root | parent | next [–]

Maybe they didn't think of it, but they certainly made it happen! The fact that they provide customizable 3D-printable models in a GitHub repo is fantastic, I love the laptop and the company.

I don't really want to bother selling it, but I'll upload it to Printables in case anyone wants it.

▲ illiac786 on Dec 11, 2022 | root | parent | prev | next [–]

But what are your steps if you loose device1? Log into each service with device2 and unenroll device1's key?

▲ stavros on Dec 11, 2022 | root | parent | next [–]

Yes, or, more likely, just do nothing because it needs a PIN that nobody else has.

▲ lokar on Dec 10, 2022 | root | parent | prev | next [–]

Yes, for each important service. Primary email, financial, dns, etc. others I user long random passwords.

▲ MetaWhirledPeas on Dec 10, 2022 | parent | prev | next [–]

Do you need it on *every* login? Isn't it like every other 2FA where you can choose "remember this device" or something like that?

▲ fmajid on Dec 10, 2022 | prev | next [–]

If a passkey can be cloned via cloud authentication, it can also be cloned when a government compels the cloud service to "authenticate" your friendly neighborhood secret policeman as being you. That in itself is a good reason to only use uncloneable credentials like a Yubikey for sensitive uses.

▲ TacticalCoder on Dec 10, 2022 | parent | next [–]

> That in itself is a good reason to only use uncloneable credentials like a Yubikey for sensitive uses.

Yup I really don't get this. For once we had something that made sense: an actual 2FA which required an actual physical device.

And now with the alliance between Microsoft (which we know can always be trusted), Google an Apple the 2FA not only is becoming "1FA" (because the goal is to use the passkey *instead* of the password, not in addition to it).

But transforming the 2FA to 1FA is not enough: we cannot let these secure hardware module make live hard for law enforcement (uh, sorry, for bad guys) so instead of having *"no device, no login"* we get instead "copyable passkey".

"1FA copyable passkey"

What a concept.

Somehow I feel this isn't exactly going forward security wise.

Then people will say: *"but it's easy. but it's convenient. And moreover it's pushed by our beloved Microsoft!"*.

Yup, exactly.

▲ simonjgreen on Dec 10, 2022 | prev | next [–]

What I see here is Yubico pushing for continued relevance while vendors, such as Microsoft and Apple, move to hybrid software hardware solutions with greater recoverability and without having to carry another piece of hardware with you. I applaud Yubico for what they've done over the years championing hardware multifactor, improved awareness of multifactor, and further in standards and acceptance. However, unless they pivot their model towards the OEM market and start selling there chips to phone manufacturers, I can't see them ever hitting the mainstream.

Both Windows Hello and Microsoft Authenticator (in passwordless mode) are brilliant examples of passwordless done in a secure way that is also incredibly user friendly to both use and set up.

▲ dwaite on Dec 10, 2022 | parent | next [–]

> What I see here is Yubico pushing for continued relevance while vendors, such as Microsoft and Apple, move to hybrid software hardware solutions with greater recoverability and without having to carry another piece of hardware with you. I applaud Yubico for what they've done over the years championing hardware multifactor, improved awareness of multifactor, and further in standards and acceptance.

One important aspect is that Web Authentication is an abstraction API for talking to an authenticator. The authenticators themselves can have wildly different policy/characteristics.

Apple in particular is pushing exclusively toward the consumer space requirements with their current implementation of Passkeys, solely as a more secure password replacement.

This means that clone ability and sharability are fundamental concepts for them, which currently go against business and regulatory requirements in several spaces. Since mandating creation of hardware-bound credentials goes against consumer usability, there isn't a clear path toward supporting them outside enterprise enablement via MDM.

Android's current plans are to only support hardware-bound credentials for second-factor usage, not for passwordless usage.

Microsoft will likely allow for user choice as to whether a credential is hardware-bound. I'm not aware if they will allow the relying party to dictate a requirement here.

Yubico meanwhile has a FIPS-certified key, and has been pushing toward support at higher assurance levels per NIST 800-63. While additional hardware is annoying for users, to the business it amounts to a per-seat cost to solve a real problem.

> However, unless they pivot their model towards the OEM market and start selling there chips to phone manufacturers, I can't see them ever hitting the mainstream.

They have an interesting model in that, while a business may need a user to have a Yubikey or similar authenticator, there often isn't a requirement that they have one specifically provisioned and shipped by the business. I've seen environments where key fobs are provisioned to users primarily through Amazon.

So I suspect Yubikeys will settle into a business user and prosumer space, but no the chance of widespread consumer adoption was always slim. The broader adoption of WebAuthn gets however, the more opportunities I think Yubico will have to sell into the gaps.

[Disclaimers: WebAuthn WG member, FIDO 2 TWG vice-chair]

▲ amluto on Dec 10, 2022 | prev | next [–]

> But passkeys aren’t a new thing. It’s just a new name starting to be used for WebAuthn/FIDO2 credentials that enable fully passwordless experiences. These types of credentials are also called discoverable credentials, or sometimes resident credentials.

Can someone explain what discoverable credentials have to do with a passwordless experience? Accepting a WebAuthn credential in lieu of a password is a policy decision and is every bit as possible with non-discoverable credentials. As I see it, discoverable credentials enable a *usernameless* experience, which is a rather different phenomenon.

▲ dwaite on Dec 10, 2022 | parent | next [–]

> Can someone explain what discoverable credentials have to do with a passwordless experience? Accepting a WebAuthn credential in lieu of a password is a policy decision and is every bit as possible with non-discoverable credentials. As I see it, discoverable credentials enable a usernameless experience, which is a rather different phenomenon.

Depends on how much information you want to leak about user accounts.

A non-discoverable credential requires a credential handle value from a previously registered credential to be given in the WebAuthn "get" request. That means you need to understand who the user is to look up the set of possible credential handles to supply.

Credential handles are a vendor-specific format and are not required to be cryptographically random, so this leaks information such as that an account is valid, how many keys they have registered, and potentially which vendor/model keys those are.

The other challenge is somewhat orthogonal to discoverable credentials, and that is authenticator support for user verification. Since the older U2F keys support neither discoverability nor user verification, and since newer discoverable credential usage doesn't make a lot of sense without user verification, the two features sometimes conceptually get bundled together.

Without user verification, your authentication only represents (at most) a possession factor. In several spaces this is considered less secure than a password and insufficient.

I've heard of efforts to support non-discoverable first-factor authentication via an initial username prompt, but I haven't actually interacted with one in production yet.

I believe I have interacted with sites which will do periodic non-discoverable challenges after an initial (password-backed) user login, e.g. step-up a long-term cookie-backed session. These are typically relying on mobile phone/desktop support, and represent it as a faster login option on that device for the future. Since these environments are getting passkeys, I suspect they will eventually migrate over to discoverable credentials with user verification.

▲ arianvanp on Dec 10, 2022 | parent | prev | next [–]

You are correct.

▲ amluto on Dec 10, 2022 | root | parent | next [–]

I have to admit I don’t really see the point of making this be part of a secure token. The “username” store (actual username, tuple of (username, FIDO blob) or whatever) doesn’t seem terribly sensitive from a local attack perspective, but it *is* fairly sensitive from a privacy perspective. Wouldn’t it work better to have this be stored by a browser, per container, etc?

Also, how is enrollment of an attested-but-not-present token or a multisig group of tokens or *anything* that enables off site storage of a token not part of the spec? It even seems like a company like Yubico could hack up a pair of tokens that separate enrollment and authentication without a spec change. Of course, discoverable credentials are a bit of a step backwards in this regard.

▲ dwaite on Dec 10, 2022 | root | parent | next [–]

> I have to admit I don’t really see the point of making this be part of a secure token. The “username” store (actual username, tuple of (username, FIDO blob) or whatever) doesn’t seem terribly sensitive from a local attack perspective, but it is fairly sensitive from a privacy perspective. Wouldn’t it work better to have this be stored by a browser, per container, etc?

An agent that registers non-discoverable credentials as discoverable ones via local storage is an option, although not one that browsers have yet chosen to support.

This however locks you onto a browser (if it doesn't have a cloud sync fabric) or into a particular ecosystem (if that browser has a sync fabric). Authenticators support discoverability because a limitation of only being able to authenticate in a single browser is significant.

Since authenticators tend to either support both discoverability and user verification, or neither discoverability nor user verification, I suspect there won't be business drivers to support such user agent storage/functionality.

Note that the CredProtect extension protects discoverability of a resident credential without user verification, and Chrome requests this extension on sites' behalf by default. This protects against scenarios where a third party who gets physical access to your authenticator (thief, partner, law enforcement, border control) can introspect the websites you have accounts at without your participation.

> Also, how is enrollment of an attested-but-not-present token or a multisig group of tokens or anything that enables off site storage of a token not part of the spec? It even seems like a company like Yubico could hack up a pair of tokens that separate enrollment and authentication without a spec change. Of course, discoverable credentials are a bit of a step backwards in this regard.

There was a serious proposal: https://github.com/Yubico/webauthn-recovery-extension

The challenge is that support for such things (new multisig algorithms, this recovery extension) require active relying party participation, and many would just choose not to take on the extra effort.

So instead we have multi-device credentials, where there is a sync fabric behind the scenes. Nothing would preclude hardware credentials from participating in such a thing, although they would obviously need either radios or software assistance to do so.

To capture the change in physical hardware, a new extension (devicePubKey) is being proposed under Web Authentication. This would have the benefit over the previous recoverability proposal in that sites opt-in to extra responsibility as needed for their business logic, compared to usability being restricted unless sites do extra work.

▲ wordlessecho on Dec 10, 2022 | prev | next [–]

It it still a problem that backup your passkeys for other platform. Android seems allow you to store passkeys in third-party provider in the future. (https://developers.google.com/identity/passkeys/supported-en...) On Apple you can only share passkey between the Apple devices. (https://developer.apple.com/videos/play/wwdc2022/10092/?time...) On Windows you cannot even manage your passkeys. But you can add multiple passkey for single account.

It might confuse people that passkeys is design as "a password replacement" (https://fidoalliance.org/passkeys) but it also can be a FIDO2 as additional authorization method. Once you setup the FIDO2 PIN for YubiKey, you have to enter the PIN every time at registering your key. Which makes me annoying and reset my YubiKey to delete the PIN.

Although Windows, Android, iOS and macOS all implement the passkey. But Google and Apple account does not support passwordless login. Microsoft still using their own implement of passwordless login. Recently Apple announce that they will support to use hardware key for two-factor authorization. (https://www.apple.com/newsroom/2022/12/apple-advances-user-s...) It looks like Apple have no interest at either passwordless login and supporting cross-platform passkey for Apple ID.

In the specification, it provides multiple algorithm choices for website developers. (https://www.iana.org/assignments/cose/cose.xhtml#algorithms) But Android, iOS and macOS only support ES256. Newer Windows support ES256 and RS256. Older Windows only support RS256. EdDSA is only supported by YubiKey.

▲ dwaite on Dec 10, 2022 | parent | next [–]

> It might confuse people that passkeys is design as "a password replacement" (https://fidoalliance.org/passkeys) but it also can be a FIDO2 as additional authorization method. Once you setup the FIDO2 PIN for YubiKey, you have to enter the PIN every time at registering your key. Which makes me annoying and reset my YubiKey to delete the PIN.

Passkeys are used to describe an alternative to passwords, and as such the term is used to indicate particular characteristics for the user (such as being a primary factor and supporting password-manager-style user experience).

FIDO authenticators and the WebAuthn API can be used to support other scenarios, such as second-factor usage. They also may support additional extensions, or support supplying attestations.

But if you wanted to say what a passkey is as a technically defined object, it is a multi-device (backup-capable) WebAuthN credential which is discoverable and supports user verification.

Likewise, there are single-device passkeys, which are non-backup-capable WebAuthN credentials which are discoverable and support user verification. Yubikeys support creating such credentials, and thus support creating single-device passkeys.

> It looks like Apple have no interest at either passwordless login and supporting cross-platform passkey for Apple ID.

Apple is notorious for not talking about future plans; however, all of the companies you mentioned are likely going to implement this technology in an incremental fashion. For example, that Apple is rolling out support for authentication via FIDO security key fobs as an advanced security option does not preclude broader WebAuthn login support for web and native access to all Apple accounts in the future as an alternative. Advanced security modes are just easier, as they are advertised as having significant usability impact.

There are just real issues here left to solve, for example authenticating to an account on an Apple TV unit, where neither USB nor NFC are generally available.

▲ marban on Dec 10, 2022 | prev | next [–]

The whole new auth methods wave is basically the equivalent of USB-C for the average consumer. Maximum confusion right from the start. Great.

▲ brookst on Dec 10, 2022 | parent | next [–]

Great analogy. Tech people love it because it’s better, and they understand the nuances and are used to having to think about things like 2D matrices of features.

Normal people expect simplicity and consistency and are totally confused.

▲ rainsford on Dec 10, 2022 | root | parent | next [–]

Speaking as a tech person, I think usability matters (or at least should matter) to tech people too. Maybe I'm just getting old, but I increasingly find that having cool features or security improvements isn't enough to make me like a product and that well designed usability is itself a tech feature worth admiring.

Most people like when technology is easy to use, but tech people can also appreciate when technology is a well designed complete package that makes it so the most people can benefit from the cool tech improvements. My favorite example here is Signal, which both significantly improved upon the security of previous solutions like PGP encrypted email *and* was a massive usability improvement such average people could take advantage of the security it offered. As a tech person, I appreciate the latter at least as much if not more than the former. Security keys check the first box, but I'm not sure they've got the usability thing down yet.

▲ brookst on Dec 10, 2022 | root | parent | next [–]

Oh, I agree. I recently purged my house of any USB-C cable that can't charge my laptop. That means no long high speed cables. But knowing that power will always work is more important than maximizing transfer speeds for my applications.

▲ sebzim4500 on Dec 10, 2022 | parent | prev | next [–]

From my experience, non tech people think USB-C is great but only because you can plug it in upside down.

▲ j6zauas4gz on Dec 10, 2022 | root | parent | next [–]

but... that is why its great? That's the only relevant factor in my day to day life anyways. that with my USB-C headphones I dont have to think about whether its upside down or not.

...am I not a tech person? Hold on, need to go do some existential musing.

▲ BoppreH on Dec 10, 2022 | prev | next [–]

I still see problems with recovery. You either have to (a) carry your primary and backup devices with you and risk losing both; (b) lower your security by using non-hardware-protected keys; or (c) be unable to create new accounts on-the-go.

What I would like is a recovery key that I can store in a rarely-accessed safe.

Here's an idea from an old paper of mine: on every sign up, you register not only your primary device, but also a *recovery pair* made of [hash(recovery_token), encrypt(master_public_key, recovery_token)]. Then you keep your single master_private_key in a safe.

If you ever lose your primary device, you then:

1. Take your master_private_key from the safe (e.g., printed QR code, backup device, passphrase for key derivation, etc).

2. Ask the service for your encrypted recovery_token and decrypt it with your master_private_key.

3. Ask the service to reset your credentials and use the recovery_token as authentication, to be checked against the expected hash.

This way you can create recoverable accounts without carrying your backup key, without downgrading to non-hardware security, without giving personal information like email address for out-of-band resets, and without creating a backup that has to be updated.

As a bonus, you can use hash(master_public_key || service_domain) as username for anonymity without extra storage. And this process also works to recover compromised accounts, something you don't get with two Yubikeys that can remove each other.

▲ sebk on Dec 10, 2022 | parent | next [–]

Your idea is not very different from this proposed spec also defined by Yubico: https://github.com/Yubico/webauthn-recovery-extension In my opinion, along with pluggable Passkey providers, this is the missing piece for a usable and secure passwordless ecosystem, especially now that iCloud will start supporting security keys.

▲ BoppreH on Dec 10, 2022 | root | parent | next [–]

> Your idea is not very different from this proposed spec also defined by Yubico: https://github.com/Yubico/webauthn-recovery-extension

Good point, in that they are both based on backup credentials generated on-demand. But the Yubico scheme is a lot more complicated and limited to backup *devices* (as opposed to passphrase or printed QR code). On the other hand, the signed handover is really cool, and doesn't require the server to store encrypted data like in mine.

> In my opinion, along with pluggable Passkey providers, this is the missing piece for a usable and secure passwordless ecosystem, especially now that iCloud will start supporting security keys.

I agree completely.

▲ adrianmonk on Dec 10, 2022 | parent | prev | next [–]

I'm not a security expert. What's the advantage of creating the recovery_token at registration time? Why can't the site just store master_public_key, and if the user ever needs to do recovery, then use it to challenge them?

▲ BoppreH on Dec 10, 2022 | root | parent | next [–]

Just for privacy, otherwise the public master key acts as a unique personal id. While there'd be no reason to publish this information, the service can still get hacked, sell your data, or just have a parent company looking for accounts in common among their services.

▲ cmdli on Dec 10, 2022 | prev | next [–]

To those who are interested in passkeys but are worried about vendor lock-in and lack of control, I would like to plug my solution, Bulwark Passkey (https://bulwark.id). It's an open source passkey manager that allows you to export your credentials outside the system, giving you more control. I think passkeys have a lot of potential to improve security and ease-of-use (passwords are broken!), but we need to maintain users' control over their data if that is going to happen.

▲ okhuman on Dec 10, 2022 | prev | next [–]

Check out AuthCompanion, a user management server that brings passkeys to your users and is pretty straightforward to setup.

https://github.com/authcompanion/authcompanion2

▲ crote on Dec 10, 2022 | prev | next [–]

I am very bearish about the current push to passkeys.

To start with, why throw out the password? We have literally spend a decade getting people to use 2FA - and rightfully so. Why would we suddenly voluntarily compromise security and go back to 1FA? Sure, people suck at passwords, but even a poor password combined with an authenticator would be better than solely an authenticator. I am quite worried that websites will adopt this form of 1FA as the "new normal", compromising the security of anyone who still wishes to use secure passwords.

Second, it looks like a blatant attempt to vendor-lock to me. The ability to sync passkeys across multiple devices is touted as a core feature, but meanwhile vendors only allow syncing between their own devices. This makes a multi-vendor environment extremely challenging to use, not to mention that switching to another vendor risks losing access to a lot of websites. And that's even ignoring the possibility of a vendor getting mad at you and deciding to nuke your account - and all the passkeys with it. Until there is a proper (open!) standard for exporting and importing passkeys, this is an unacceptable risk to me.

Some of this could be mitigated by a universal pluggable passkeys backend, but I am not really seeing any substantial movement in this direction. To me, it looks like Google/Apple are trying to replace "Log in with Google/Apple" with "Log in with Passkeys" to make it *seem* like it is an open platform, while still locking it down as tightly as possible. Despite being an open standard, the implementations are anything but.

Meanwhile, syncing provides a security risk. Every synced device has access to all websites all of the time. If you lose sight of a device, an attacker also has access to those websites - and they might even clone the passkeys. This opens you up to attacks which are extremely hard to detect, and due to the passkey being shared between all devices also relatively difficult to stop.

I am also seeing a lot of misleading statements surrounding its security. For example, the claims that the keys are stored in a TPM/Secure Enclave. Meanwhile, in practice the TPM is only used for the *local* encryption of *a copy* of your passkeys. They also happily live on the cloud because they are being synced, and the vendors seem to be offering ways to recover them without access to a *single* device. They seem to be intentionally reusing language traditionally reserved for security implementations where the private key is stored solely in a HSM with no way to export it, but that is not at all what is happening here. Security-wise, it's nothing more than a rebranded cloud-synced password vault.

Meanwhile, hardware token vendors like Yubikey point this out, but they do not see it as a threat as they are selling this synced implementation as not suitable for high-security environments - and rightly so. Luckily, the protocol has a built-in way to bypass this: the passkey backend can be asked for attestation, which in essence is an immutable certificate embedded in the hardware key, signed by the hardware vendor. If you want people to use a high-security passkey implementation, simply require attestation from a small list of well-known vendors - ideally only Yubikey of course. By hopping on the Passkeys bandwagon, they are able to erect significant barriers to entry for any potential competitors.

Finally, Passkeys provide a privacy risk. With traditional Webauthn the user provides a username to the server, and the server responds with a set of opaque blobs which the user's tokens use for further bidirectional authorization. With Passkeys, the server asks the token for their set of domain-relevant passkeys, which *include the username*. This is required for the intended single-click usernameless login, but the flip side is that an attacker would be able to discover all of your accounts by probing the token. Alternatively, the server would be able to discover an association between multiple accounts of the same person, destroying attempts to retain an anonymous alt account.

All in all, to me Passkeys looks like a thinly-veiled attempt to lock users into certain ecosystems, using convenience as a carrot-on-a-stick. Meanwhile, it adds significant security and privacy risks while *claiming* to be more secure. Seeing this being touted as a "more secure password replacement" by such major players makes me weep for the future.

▲ dwaite on Dec 10, 2022 | parent | next [–]

> Why would we suddenly voluntarily compromise security and go back to 1FA?

Nothing about passkeys precludes you from doing additional security checks.

The challenge is that once you say "but an iCloud-synced passkey is not secure enough", there are very few options which are better - specifically because the attacker model requires either user participation or both an iCloud and existing local device passcode compromise.

Many sites are relying on SMS, TOTP, or email second-factor challenges, when the attacker already would have more than enough information to clone the TOTP from backup, to port the SMS number over (or just get the same push on a new device they control), or to read your email.

So unless you are doing in-person verification or relying on a secondary hardware-bound factor, there's limited "strength" utility in additional checks.

We are generally hitting the edges of what the multi-factor model can represent. All models have limitations. My opinion in particular is that MFA is meant to represent an organization's challenges for authentication factors, rather than requests challenging against an externalized authentication process with its own business logic.

> I am quite worried that websites will adopt this form of 1FA as the "new normal", compromising the security of anyone who still wishes to use secure passwords.

We can only hope. This is both more secure and more usable than what consumer facing websites do today, including many banks, from a user perspective.

The challenge is that an externalized authentication process will not map into risk mitigation processes or regulatory requirements today.

I suspect what will happen is that there will be a broad rollout, without actual security events, in a lot of verticals, and the statistics from that (e.g. phishing on password+SMS vs passkey logins) will motivate more productive reasoning on how new verticals can accept it.

> Sure, people suck at passwords, but even a poor password combined with an authenticator would be better than solely an authenticator.

Passkey-capable authenticators can do user verification, such as with a PIN or biometric. User verification is the expected mode of operation for passkeys. The challenges are that the site is abstracted away from understanding what steps were taken, and that a multi-device credential does not meet strong physical possession requirements.

Speaking broadly, there is a conflict between supplying a site with more control over the options which may or may not be supported in an authenticator, vs such control leading to sub-par user experience and even rejecting the user's choice of authenticator. I don't think anything will resolve this conflict other than time changing people's perceptions on what they need.

> Second, it looks like a blatant attempt to vendor-lock to me. The ability to sync passkeys across multiple devices is touted as a core feature, but meanwhile vendors only allow syncing between their own devices.

There are capabilities to use devices across ecosystems. Generally this is done via a proximity-backed QR code initiated flow.

The expectation is that sites will allow for multiple registered passkeys against a single account, and that sites will be motivated to register an additional passkey for future local login if such a cross-device flow was used.

> Some of this could be mitigated by a universal pluggable passkeys backend, but I am not really seeing any substantial movement in this direction.

Android has publicly announced future plans to support such a thing, although I don't believe they have announced an approach or timeline.

1Password and Dashlane have both announced their web extensions will allow them to operate as passkey providers, and I believe both have now shipped at least early access code. I would certainly say the 'interest' is there to support pluggability.

> Meanwhile, syncing provides a security risk. Every synced device has access to all websites all of the time. If you lose sight of a device, an attacker also has access to those websites - and they might even clone the passkeys. This opens you up to attacks which are extremely hard to detect, and due to the passkey being shared between all devices also relatively difficult to stop.

How is this different than the state of password managers over the last decade? Both passwords and passkeys require local multi factor (possession of the mobile device plus a passphrase/biometric) to do any of these actions.

> I am also seeing a lot of misleading statements surrounding its security. For example, the claims that the keys are stored in a TPM/Secure Enclave. Meanwhile, in practice the TPM is only used for the local encryption of a copy of your passkeys.

In practice, it is quite a bit more complicated, including being specific to the device or software ecosystem.

On Apple devices, in addition to normal HSM clustering-style provisioning of new devices, there is support for a user-controlled paper recovery key. Like most implementations of cloud-synced vaults we've seen over the last decade, the vendor has no capability to read the data of the vault.

From an organizational risk perspective, there's no way you can legitimately accept passkeys when you can't evaluate their authentication policies or backing security.

From a user's risk perspective, they are 100% correct in assuming this is a secure system.

> If you want people to use a high-security passkey implementation, simply require attestation from a small list of well-known vendors - ideally only Yubikey of course.

A workforce can do this by requiring the keys they provision, but it is not am acceptable course of action for a public-facing domain. See https://www.chromium.org/security-keys/ :

"To this end, public websites that restrict the set of allowed Security Keys should do so based on articulable, technical considerations." "If Chrome becomes aware that websites are not meeting these expectations, we may withdraw Security Key privileges from those sites. "

You must map your policy for what are acceptable security keys onto the _ecosystem_ of available attested security keys. Just accepting Yubikeys may eventually cause your site to lose access to WebAuthn API in certain browsers.

My understanding is that this is done so that early movers in the security key space (like a Yubico, who has proposed, co-authored or contributed to most of these specifications) don't gain market lock-in.

> This is required for the intended single-click usernameless login, but the flip side is that an attacker would be able to discover all of your accounts by probing the token.

The Web Authentication API does not provide a way to introspect which authenticators are available or which credentials they may store. Instead, the user is given some consent interface by which to release one credential on one authenticator.

Since credentials are also origin-bound, the only difference is that an attacker with malicious code on that origin could get you to release a credential without phishing any previous steps (e.g., asking you for your username and first-factor password login, so that it can see valid credential handles).

For CTAP, there are two protections. CredProtect prevents enumeration of credentials without user verification, so someone who gets your key cannot see the domain you have accounts on without further interaction from the key fob owner. Second, platforms have generally been locking down the ability for applications to directly interact with CTAP 2 interfaces, possibly behind an entitlement.

▲ zamadatix on Dec 10, 2022 | parent | prev | next [–]

The below adds my understanding, which may very well be horribly wrong, of things. Corrections welcome as I try to figure this out myself!

> We have literally spend a decade getting people to use 2FA

This still supports MFA it just also supports SFA. Requiring MFA 100% of the time is a guarantee for something people won't use even if it fits security ideals better.

> Second, it looks like a blatant attempt to vendor-lock to me. The ability to sync passkeys across multiple devices is touted as a core feature

This one isn't as clean as I like either. There is a cross-device authentication flow, you can sign up multiple authenticators, and yubico has a backup authenticator proposal but none of these are necessarily ideal. I get not wanting to make the private key exportable for security reasons but if it's going to be allowed to sync on software authenticators over the internet I'm with you it should be able to sync between providers too. I'm not saying everyone should have to use cross-vendor sync I'm just saying it should be a capability.

> Meanwhile, syncing provides a security risk. Every synced device has access to all websites all of the time. If you lose sight of a device, an attacker also has access to those websites - and they might even clone the passkeys. This opens you up to attacks which are extremely hard to detect, and due to the passkey being shared between all devices also relatively difficult to stop.

You don't have to use external sync if you don't want and the standard still supports MFA on the passkey if you want but the truth is any solution that forces both of these 100% of the time will never see mass adoption.

> I am also seeing a lot of misleading statements surrounding its security.

I think a lot of it has to do with there being different levels of passkey store for different use cases. John Doe may want the convenience of just setting up a key on his phone and having it sync to all his devices for personal stuff but Jane Doe may need a hardware key with MFA for accessing the code repo at work and every option in-between. I don't think the security promises are misleading just maybe more complicated than one would wish because it's trying to be 1 way to cover all users and use cases not just "secure way" or "easy way for 1 particular device".

> Meanwhile, hardware token vendors like Yubikey point this out, but they do not see it as a threat as they are selling this synced implementation as not suitable for high-security environments - and rightly so. Luckily, the protocol has a built-in way to bypass this: the passkey backend can be asked for attestation, which in essence is an immutable certificate embedded in the hardware key, signed by the hardware vendor. If you want people to use a high-security passkey implementation, simply require attestation from a small list of well-known vendors - ideally only Yubikey of course. By hopping on the Passkeys bandwagon, they are able to erect significant barriers to entry for any potential competitors.

I don't know about "they are able to erect significant barriers to entry for any potential competitors" as the open passkey standard is specifically part of the next generation hardware key standard (FIDO2) rather than get locked into a proprietary hardware passkey format but I do agree it's the high-security way to use passkeys (as highlighted in my comments above). Not being able to export the private key, while annoying/disappointing in the cloud sync use case, is a hard requirement for the high-security use case so isn't an example of being there to provide lock-in. Especially with the ability to register multiple hardware authenticators for the same account.

> Finally, Passkeys provide a privacy risk. With traditional Webauthn the user provides a username to the server, and the server responds with a set of opaque blobs which the user's tokens use for further bidirectional authorization. With Passkeys, the server asks the token for their set of domain-relevant passkeys, which include the username.

I don't think the response contains any user identifying information. It does contain an opaque identifier the relying party can use to map to which user it was called the user-handle but that's not the user name it's a 64 byte sequence that can be mapped to a username if the other side already has it.

▲ butz on Dec 10, 2022 | prev [–]

Could anyone comment on recovery options? If Yubikey, smartphone or computer breaks, how do I login?

▲ cuu508 on Dec 10, 2022 | parent [–]

You use your second key.