Position Paper on Authentication

Eric Grosse

2022-01-24

1 Context

The federal government and OMB in particular have embarked on an ambitious and focused effort[5] to deploy phishing-resistant MFA and management of trusted devices as top priorities.

Let us assume a *competent defender* such as a federal agency with secrets to protect and staffed with an infosec team qualified to do so. As one indication of competence, suppose the agency computers are already kept patched up to date and logs of security-relevant events are archived and analyzed for evidence of compromise. Let us also assume a *competent adversary* such as GRU or MSS with substantial resources and an approved espionage mission. (No moral judgment here. All the advice is unchanged if the defender is a Russian federal agency and the adversary is NSA and GCHQ.)

This position paper tries to give advice relevant to the OMB priorities while meeting the standard of "new, true, and interesting." It may be provocative in places but I believe it is all well justified from practical experience fighting off these adversaries. I'm writing informally for an audience that I assume already knows a lot about technical security but might not have yet collected a decade of experience. These are my personal views, which the reader is expected to combine and contrast with others' to decide on their own set of best practices. In the aviation community we have a saying "learn from others' mistakes because you won't live long enough to make them all yourself."

2 Security Keys

For years we engaged in an arms race with the adversary over passwords and various forms of second factor. Collecting login credentials (both individual and administrator for computers, networks, and applications) was observed to be a primary attack vector because of the extraordinary access it provided. This arms race came to a sudden stop when Security Keys[9] were introduced.

The essence of phishing is that a user is tricked into visiting a login page under control of the adversary. (*User* here may mean an agency employee or a citizen client of the agency.) The solution is not to train users to avoid being tricked into visiting the login page; there are an unbounded number of ways for

the adversary to achieve this beyond the control of the agency or the user. Often it can be achieved in a context that distracts the user and implies urgency. I have personally seen very paranoid, very astute security experts fall for this, so don't even attempt the phishing education approach because you're just wasting your time and your users'.

The fake login page can obviously collect and replay a password but it can also collect and replay any six-digit-code or similar from SMS or hardware OTP token or smartphone app. So don't introduce such things in hope of "raising the bar" or taking "first steps." That is expensive in user productivity, in system roll-out costs, and most important in your team's future credibility. Our goal here is not to take the next incremental step in an arms race but rather to take such a large step that the adversary has to switch to an entirely different attack path.

To the extent that passwords are still used, try Password Authenticated Key Exchange[7] so that even full compromise of the server does not give the the adversary any useful password clues. But passwords still have problems when used alone, so let's keep going.

Security Keys (using the open standard FIDO U2F, now part of WebAuthn) are one of the top available phishing-resistant MFA[4] solutions because in the original and best use they partner with the browser to include the url of the login page as part of the cryptographic exchange. Visual similarity is no longer enough to trick the user. It is difficult to overstate what a remarkable effect this had stopping cold all account compromise, as observed at Gmail for ten years now.

Agencies might decide to adopt PIV or CAC cards instead of Security Keys because of sunk investment in employee distribution or trust in the hardware supply chain or fear of USB ports. That is fine as long as the essential lessons have been learned. First, the URL of the (server-TLS-authenticated) login page needs to be part of the cryptographic exchange. Second, the hardware needs to be usable on *all* devices that agency users may need to use for work, and must not require extraordinary effort installing middleware drivers.

Not everything is a web page, so WebAuthn may not apply for all agency needs. The recent OpenSSH innovation of Security-Key-enabled secret keys can point a way forward, though a skilled agency cryptographer should review any such new designs.

That's the good news, that as an industry we've invented and proven out in large deployments that we can make decisive progress against what used to be the most important attack vector. For the bad news, let's turn now to the various practical problems of deploying Security Keys.

The Achilles heel of most strong authentication is account recovery. People do lose physical objects, just as they forget passwords, so you *have* to have a sound plan for how to restore access. Adversaries know this and are very skilled at social engineering, so put a lot of thought into your design. It may suffice to require physical presence at a help desk where cross checking of the person against an employee photo database and logs of revoked Security Keys can be done. In a remote work environment, perhaps a conference video call of the

help team with the employee and the employee's supervisor and physical mail to an address of record is an adequate substitute for physical presence, though the quality of the best deepfakes does give me pause. You may already have a solid process in place for lost building access badges that you can adapt, but keep in mind the skill of the adversary and all the corner cases in a globally dispersed employee population. Stress test your design by confirming that even your agency CEO can't override the rules.

It is easier to solve the account recovery problem for employees than for client users. For the general population your best option may be to let the user decide for themselves how much security to opt in for, at the risk of getting locked out of their account if sloppy. Give them good explanations so they can make an informed decision.

An unfortunately common way to evade dealing with the account recovery problem is to allow the unauthenticated user to pick an alternative second factor at time of login, but this enables downgrade attacks[6] and should not be done if your system deserves strong security. As an example, for Gmail the pure form of Security Key without downgrade is called "Advanced Protection Program"[1] and I have long encouraged all my friends and family to enroll.

A pretty good solution to the problem of lost keys is having a backup Security Key in a safe location. Make it easy and natural for the user to become aware of a rarely used Security Key coming back to life. Don't try to economize; pay for your employees to have two or more keys and allow your clients to register and name multiple keys. Speaking of cost, understand that the manufacturing cost of a Security Key is on the order of a few dollars, so resist letting vendor greed price you out of an otherwise sound plan. But even if your vendor charges two orders of magnitude above manufacturing cost, the couple million dollars for a big deployment is almost surely cheaper than dealing with a single breach incident.

The flip side of people losing physical objects is that a thief must not have a long window for using the object. Requiring the adversary to capture both password and Security Key is one way to help with this, so I don't recommend tokens as an alternative to passwords but as a second factor. It has to be easy to identify which Security Key has been lost and quick to revoke it, even before regaining full account access. That probably means a help desk interface that allows seeing nickname assigned to the Security Keys and time of most recent use. Obviously there are some tricky privacy issues there. Careful process design needs to be done as part of the account recovery plan and, ideally, published for other agencies to replicate. We want to be careful not to introduce an easily exploited Denial of Service vulnerability.

By design, a Security Key does not actually store the per-system credentials on-token, so a single hardware token can handle an arbitrarily large number of different system logins for the user, even among organizations that do not know of each other. When you participate in this ecosystem, you're not asking your users to carry an additional object beyond what they should already carry to protect their personal Gmail or Github or Twitter or Vanguard logins. When the user leaves your organization, don't ask for the hardware token back, just

revoke the credential record in your database.

The Chrome browser is about to remove the older U2F API, requiring developers to switch to the WebAuthn API for implementing Security Keys. That is not a problem for new deployments and is mostly an issue for Google to sort out, but I mention it in case there are any old Security Key federal pilot programs that need to be checked for updates.

On a longer time scale, the agency will need to worry about security of the hardware supply chain. Obviously substitution during initial shippping of malicious devices in place of good ones wrecks the system. But even intact devices from the factory could be vulnerable if there are implementation mistakes in the cryptographic chip. In Google's first design, this chip came from NXP which has a reasonable reputation for handling vulnerability reports but we know is not perfect. We later chose to design our own hardware, now open-sourced in OpenTitan.

On an even longer time scale, since the U2F standard relies on elliptic curve cryptography, quantum computing is a possible threat. If this becomes the top remaining risk at your agency, good for you; you must be doing a lot right.

3 Delegation

Any authentication system that replaces classic password login needs to deal properly with support of delegated access. A General once told me that he was annoyed by CAC login because he needed to hand his card to his Colonel to perform some action, then would find himself locked out of the building coming back from lunch. Inside Google we had a similar problem when first deploying Security Keys; I needed to teach my fellow managers how to use Gmail's little-known delegation feature to enable their executive assistants to read and send email on their behalf without actually logging in as them.

In short: be sure all your application software has solid delegation capabilities before you try to enforce strong authentication, and include that in your instructions or troubleshooting guide.

For bonus points, ask if your security operations center has analysis tools that let you identify likely informal delegation via shared passwords and do a pilot trial with those users first.

4 Bad Security Policy

It used to be common security advice to change passwords frequently, independent of any specific disclosed breach. Similarly, it used to be common to insist on particular mixes of character classes in a password, which frustrates people using solid methods such as high-entropy passphrases. In practice, picking and reliably saving a good password is so much effort for the individual that a sudden forced password change perversely encourages bad practices such as appending the year to an unchanging password. Probably no competent defender would

introduce such obsolete policies today, but they may not have yet gotten around to fixing all their legacy systems, especially client-facing ones. Do it.

"Phishing education" was discussed above. Don't do it.

Don't turn your user into your adversary; impose the minimum effective rules and communicate them clearly and honestly. Authentication, as much as any part of security, benefits from crystal clear mental models in the users' heads of how it all works and how much depends on it.

5 Device authentication

Even a convenient user authentication system is burdensome to run at each screen unlock or at each command or in the most silly extreme between each character typed. In practice, it is generally regarded as a reasonable compromise to require strict authentication when a device has been out of a user's presence for an extended period but otherwise allow a weaker device-login that is accepted by remote servers as equivalent to a full user login. Thus in the example of a Gmail account protected by a Security Key on a single user device, we may only require the token to be plugged in when a user first authenticates on a new device, but then set a local login cookie in the browser that will be trusted for future mail commands. In effect, the device itself becomes a hardware authentication token.

Some enterprises set an expiration of an hour or a week or a month for such intermediate credentials. The trouble with picking a time length is that in today's semi-automated threat environment any specific period is either too long to be helpful or too short to be usable, much as we found in the bad old days of password expiration.

A well managed network of devices goes further and attempts to monitor how well the device is kept up to date with security patches and perhaps user presence.

There is a whole separate topic of cloud application authentication beyond the scope of this paper, but a rich source of massive data compromise[3]. If your agency is moving to the cloud, be sure to get expert review and automated checking of any permission changes so that mistakes are caught promptly.

It can be useful to require user-less device authentication, primarily a machine certificate that points to a central database of software update status, before connecting the device to the intranet. That is to say, don't only ask the device if it is (perhaps falsely claiming to be) patched up to date, but confirm at your patch server that the device has a history of installing patches promptly and ideally also has a known acquisition provenance. Failing that, leave the device on a guest network or perhaps a patches-only purgatory network. Under the zero trust philosophy, mere presence on the intranet is not enough to get any substantial sensitive access but network access control can be helpful anyway. One advantage is to give the agency a workable system for machine inventory. Another advantage is to improve the signal-to-noise ratio of log records that operations teams analyze, since by removing easily exploited legacy equipment,

we remove some of the routine background of network scans.

Just as with firewall-based perimeter security, the of zero trust philosophy is not to get rid of perimeters but to prevent people from depending on a weak safety net.

6 Pass-the-hash

A common technique for lateral movement by the adversary through the defenders network has historically been to leverage local compromise of a device to collect intermediate credentials such as the local login cookie mentioned above or a LANMAN / NTLM password hash, which can be replayed directly in a low-level protocol to get access to off-device resources, a method known as pass-the-hash[8].

There were commendable efforts to bind such intermediate authentication credentials to better protected secrets such as TLS session state. As far as I know, the state of art is not yet mature enough to recommend a specific approach. Note this in your TODO checklist as an important remaining authentication problem even after strong user authentication such as Security Key has been fully deployed.

7 Rubber hose cryptanalysis

I wish I knew how to design an authentication system that could protect against a misbehaving roommate, but alas I do not. Dishonest or abusive partners are a real world problem and fortunately there is a growing security literature on this, but there are few strong technical solutions. A related situation exists for residents or visitors of authoritarian countries.

From what little I know, the best you can do is offer a deadman switch, under which if the authentication system is not used for a period of time stringent extra checking is done upon next use. As with duress passwords, people with more experience tell me there are severe personal risks to the user if you get the architecture wrong. So I have little actionable advice except to encourage you to talk over your design with representatives of your most vulnerable users.

8 Invitation to talk

There are many nuanced points in this paper, any one of which merits extended discussion. I hope it can be a starting point for a conversation with those who know their agencies' constraints better than I. As an example, I'll give the next step of a discussion on phishing education:

Your security team has a limited number of hours in the day, and I claim phishing education across the broad user population is not the highest and best use of their time. If a real, hopefully limited, incident at the agency occurs then it can be worthwhile for an entertaining storyteller from the security team to

brief the details at an all-hands meeting being held anyway. But the customary phishing education program sends an email and, when an employee clicks on an embedded link, scolds the employee. That's counterproductive because the same agency also has lots of official emails from human resources that include links. I agree with the agency employee who says "Oh, those security dorks live in fantasy land. Of course we click on links; that's the whole point of the web. If they don't like it, fix the browser." Besides, it's not just about clicking on suspicious links; look up "watering hole attack" sometime. My team put lots of effort into fixing the browser instead of running phishing exercises. Even more important than their limited number of hours, your security team has a limited stock of political capital. Spend it wisely.

This is not meant to be a blanket prohibition on all such exercises. In fact, my security team for fun and insight had a tradition that people switching offices or leaving the team would give permission to their teammates to have a month to do anything they want to steal their password. Of course phishing emails and chats were used, but also substitute keyboards, substitute Chromebooks, cameras in the ceiling, Linux kernel exploits, and so on. It's all great fun and no one is held up to ridicule or scolding. One of the most creative defenses involved scheduling departure from the office for one date but actually leaving a month early. That (exceptionally careful and talented) team member in later years fell for an actual phishing email, fortunately protected by other layers of our defenses, which convinced me once and forever that we're not going to solve the problem through education.

9 Thanks

These personal remarks are largely based on my experience at Google [2] and I humbly thank the brilliant people there in identity, threat intelligence, incident response, application security engineering, and systems reliability, who deserve all the credit for anything you learn from this paper. There are too many to mention individually but I do wish to single out Marius Schilder (and through him Jakob Ehrensvärd) for their Security Keys which did more than anything else to advance practical authentication. Thanks to Tho Nguyen and Mary Ellen Zurko for clarifying questions on an earlier version of this position paper.

References

[1] Google. https://landing.google.com/advancedprotection/faq/.

[2] Eric Grosse and Mayank Upadhyay. Authentication at scale. *IEEE Security & Privacy*, 11(1):15–22, January/February 2013.

[3] Aaron Haymore, Iain Smart, Viktor Gazdag, Divya Natesan, and Jennifer Fernick. 10 real-world stories of how we've compromised ci/cd

pipelines. https://research.nccgroup.com/2022/01/13/10-real-world-stories-of-how-weve-compromised-ci-cd-pipelines/.

[4] Nils Höll. Beyond passwords? about the current state of FIDO2 authentication, 2020. https://github.com/nilshoell/fido-paper/blob/master/hoell2020_fido2-auth.pdf.

[5] Office of Management and Budget. Moving the U.S. government towards zero trust cybersecurity principles, September 2021. https://zerotrust.cyber.gov/federal-zero-trust-strategy/.

[6] Enis Ulqinaku, Hala Assal, AbdelRahman Abdou, Sonia Chiasson, and Srdjan Capkun. Is real-time phishing eliminated with FIDO? social engineering downgrade attacks against FIDO protocols. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3811–3828. USENIX Association, August 2021.

[7] Wikipedia. https://en.wikipedia.org/wiki/Password-authenticated_key_agreement.

[8] Wikipedia. https://en.wikipedia.org/wiki/Pass_the_hash.

[9] Yubico. https://www.yubico.com/products/security-key/.