

Gone Phishing

Airbnb's journey to phishing resistant MFA
Graham Gilbert

> **whoami**

I'm Graham

Client Engineering at Airbnb

londonappleadmins.org.uk

macadmins.io

graham.at/movember

Hi! I'm Graham, and I'm a Senior Tech Lead Manager at Airbnb on the Client Engineering team. I am one of the co-founders of London apple admins, I am on the board for Mac admins open source, and I am also a Movember fundraiser - it was at this very conference five years ago that I kicked off my campaign and to date I've raised nearly \$40,000 thanks to the generosity of our community - so if you find this talk useful in any way, please consider donating. And today we are going to talk about Airbnb's journey to phishing resistant MFA.

Some definitions

But before we start, let's go over a couple of pieces of terminology I'll be using today

What is MFA?

- Multi-factor authentication
- Prevent access using only a password
- Something you know (e.g. password)
- And something you have (e.g. phone, security key etc)

MFA stands for Multifactor authentication

The goal is to stop someone being able to gain access using only a username and password

It consists of Something you know - for example, a password

And Something you physically have like a phone, a security key or a number generator keychain thingie

OR Something that is unique to you for example, biometrics - finger print, Face ID, retina scan etc

What is phishing?

- A type of social engineering attack
- Attacker masquerades as a trusted entity
- Goal is to convince the user to open a message or notification

Phishing is a type of social engineering attack that is often used to steal user data, including login credentials and credit card numbers. It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, text message or other notification.



IDP = Identity Provider

Finally, IDP is an Identity Provider. This is your source of truth for the user's identity. This could be Okta, OneLogin, Ping or any of the other countless identity providers.

Our story begins in 2016

I started working at airbnb in 2016 - we were at the same mfa level as most companies 7 years ago - we had an IDP and we enforced MFA on it. We didn't really have any restrictions on what kinds of MFA you could use - you could use anything, including SMS and phone



The first thing we did was disable MFA via sms and phone call

We looked at the logs, and we found that very few users were actually using sms or phone for MFA

And the vast majority of those that were using it had alternative mfa methods available to them - for example they had a smart phone so could use push via the app, so it was pretty straightforward to walk them through setting up a better form of mfa

**Why is SMS or phone based MFA
not a great idea?**

Why did we rush to turn off these forms of mfa?

**SMS is not encrypted and
vulnerable to man in the middle
attacks**

Because SMS is not encrypted, if the cellular infrastructure is compromised, it would be pretty trivial to get your message

SIM Cloning and Swapping

<https://techcrunch.com/2023/02/01/google-fi-hack-victim-had-coinbase-2fa-app-hijacked-by-hackers/>

Sim cloning is the process of copying over the IMSI (International Mobile Subscriber Number) and Authentication Key to another sim card. This allows a bad actor access to incoming calls and messages - This does usually require physical access these days though, so its less common

SIM swapping is a type of ATO (account takeover) attack

A bad actor convinces your phone provider to port your number to another SIM card

This is pretty common, and there isn't anything you can do about it this is entirely down to your phone provider performing adequate identity verification— this is the primary reason you should consider phone and sms mfa insecure.

In 2016 we also started issuing yubikeys to our engineers to store their ssh key on - this had the byproduct of kickstarting the usage of security keys at airbnb

2017: The year of the Linux desktop


Then we get to 2017

2017: The year of the ~~Linux~~ Chrome OS desktop

In 2017 we started introducing Chrome OS at our third party contact center partners

These users were only allowed to use Security Keys

However due to a limitation in our MFA provider with the Chrome OS login screen, we had to use U2F - a legacy protocol for security keys



**And that's how we
stayed for a while**

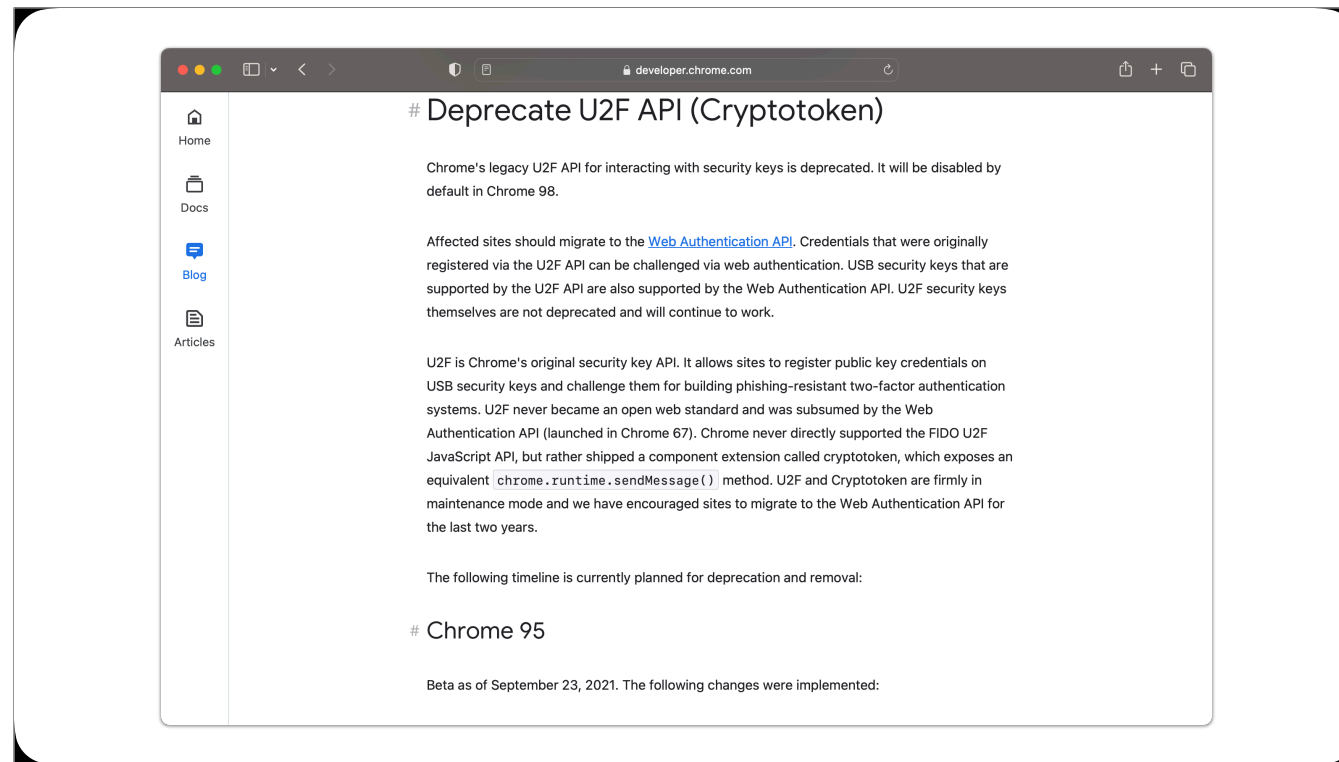
We had corporate users using the MFA vendor's app with push notifications and a security key if they had one

We did start shipping security keys to everyone who got a new laptop in late 2017, not just engineering, but we didn't enforce their use for MFA (although it was encouraged during their onboarding)



Until 2021

We'd been thinking about moving to only using security keys for a while, but our hand was forced in 2021 into making some changes.



If you recall I said we set up all of our security keys at our contact center partners using U2F. In 2021 Google announced that they were removing the U2F api from Chrome.

At this point we probably had around 15 - 20,000 chrome os devices

That's a lot.

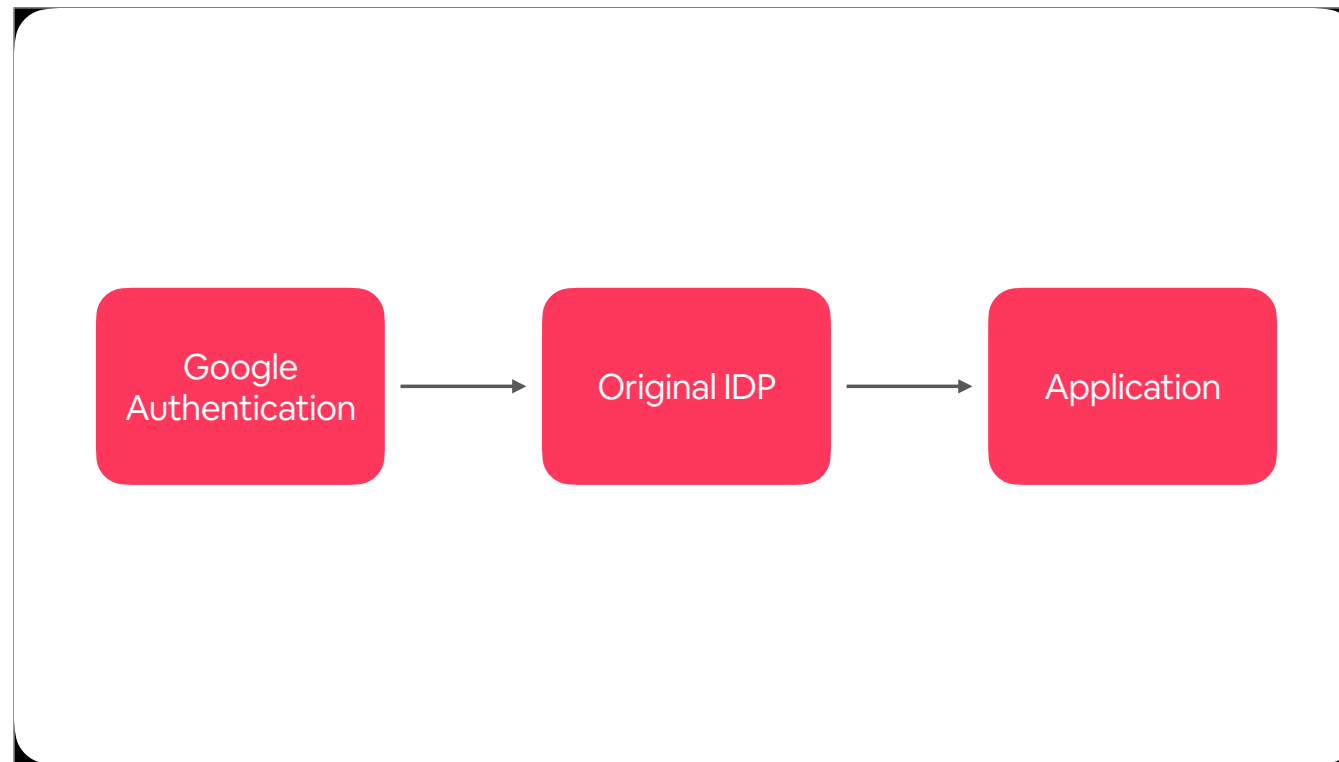
At the time, our MFA provider didn't have a migration path from U2F to Webauthn - the user needed to re-enroll their security key. So we had 20,000 users that needed to migrate and would have to do this manual process

So if they're re-enrolling their keys, we might as well make a change for the future.

we really liked the security key flow in Google - in addition to when your session expired and you needed to re-authenticate, it also kept an eye out for suspicious logins - like when you come from a different device or a different location - which allowed us to have a longer session length than we would have been able to on our previous MFA tool.

Trusted IDP

Around that time, Our IDP introduced support for something called Trusted IDP - which is using the SAML assertion from one Identity provider to authenticate a second



So in our case it meant we could set up our original IDP as a SAML application in Google, and then use that to authenticate to each application in our IDP. This meant we didn't need to reconfigure our several hundred applications in our IDP to be able to take advantage of Google's security features.

A large, solid pink rounded rectangle with the word "Webauthn" centered in white text.

Webauthn

Before going any further, I think we need to talk about Webauthn.

Webauthn

Open standard

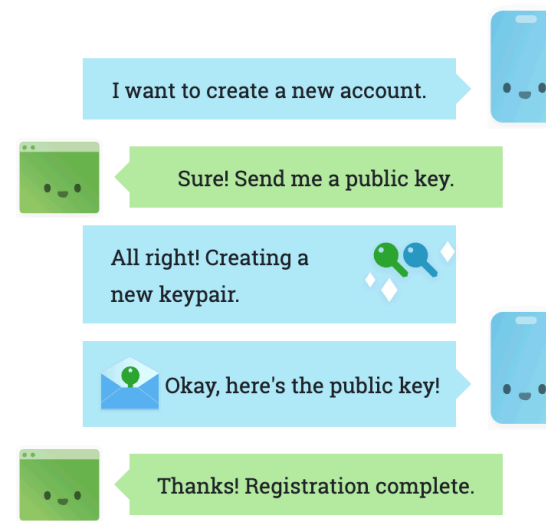
Supported in all major browsers

Register and authenticate users

Wide range of authenticators

Unlike its predecessor, U2F, Webauthn is an open standard and is supported in all major browsers. The Webauthn API allows servers to register and authenticate users, and the authenticator can be a variety of things - it could be internal to the computer using a TPM, or the secure enclave, or using an external security key such as those made by Yubico or Fethian or it could be a passkey (as long as you don't use managed Apple IDs).

Registering a Webauthn credential



<https://webauthn.guide>

So when you want to register a new credential (which could be used as either a second factor alongside a username and password, or as a primary factor in a password less login), the authenticator generates a public and private key - the private key in most cases is generated on dedicated hardware, in our case a yubikey. The authenticator sends along the public key to the server it is registering with. The domain name the credential is generated for also stored on the authenticator, so that key pair will only ever be used for that domain, and we know that the authenticator is physically present on the device that is authenticating to the server, which eliminates a big phishing attack surface.

Authenticating with a Webauthn credential



When you want to sign in with your credential, the server will send you some random data. You sign it with your private key and then send it back to be verified. An important thing to note here is that you need to physically authorize the use of your private key - this is often touching a button on your security key, or using biometrics like a fingerprint - this ensures someone is physically present, so malware shouldn't be able to authorize use of the private key.

The migration (phase 1)

We planned our migration in various phases.

Why contact center partners first?

Limited number of applications

Mostly in-house applications

Chrome OS only

Already using security keys

Access to customer data

Our first group to move onto our new authentication flow were our third party contact center partners. We chose them because they only use a small number of applications, which are mostly written in house - which means they they should already support modern authentication standards, and if any didn't, we had the ability to change them.

This group of users is exclusively using chrome os - we do this for security reasons, but it also has the side benefit of them only using a browser and potentially a couple of android apps.

This group of users has been using security keys with U2F for many years, which meant they were used to the workflow with using them, had supply chains already established for keys and none needed to be shipped, so we could move relatively quickly.

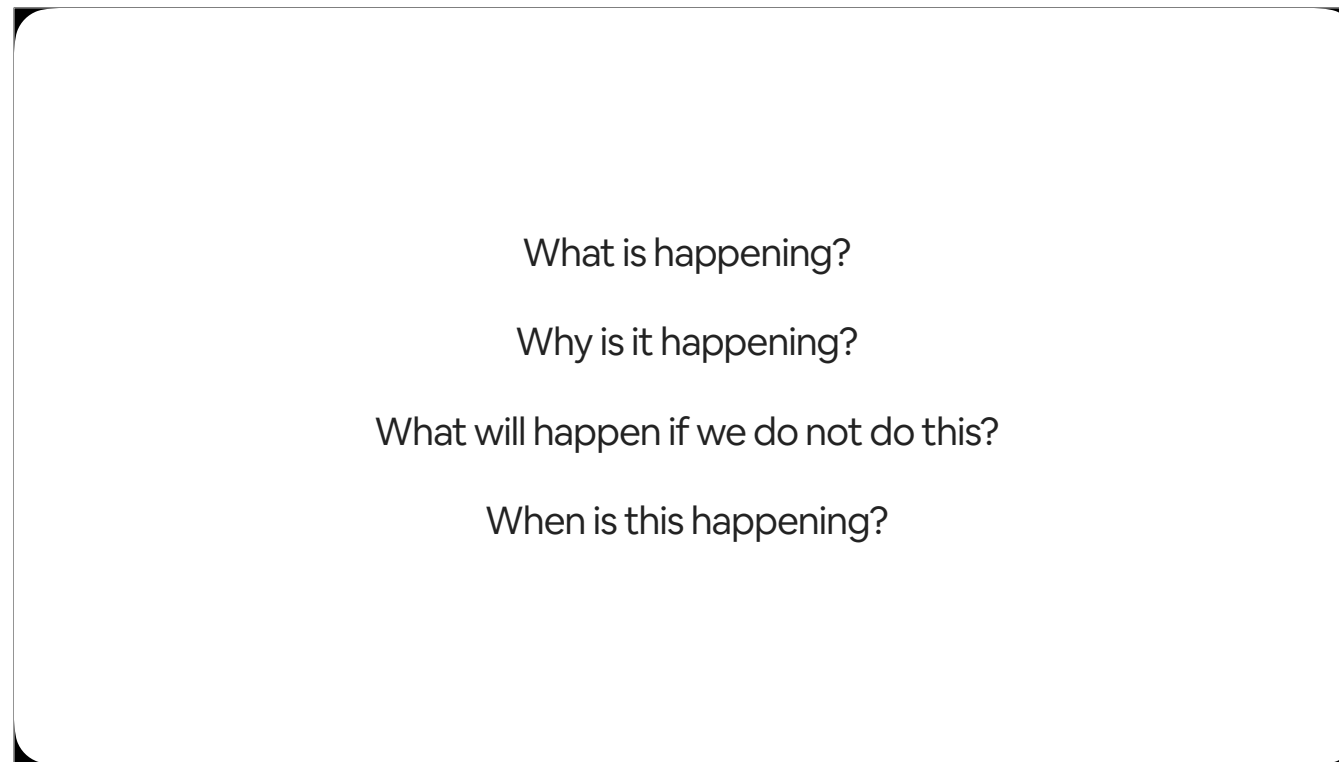
Finally, this group of users has access to our most sensitive data - our customer data.

Planning

Once we had decided who was going to be moved over first, we could get stuck into actually planning this thing out.

Getting buy in from our stakeholders

This is clearly a pretty disruptive change - we needed help from all over the business, from our liaisons at our CS partners, our own IT support folks, the people in charge of change management to name a few



There are four key pieces of information you need to give to your stakeholders to get buy in for any disruptive change:

What is happening? Be clear, avoid tech jargon. Explain what your end users are being asked to do.

Why is it happening? Once again, avoid tech jargon here. You could outline the benefits if it is an unforced change, but in our case we simply said that Google are sunseting the method in which we use security keys.

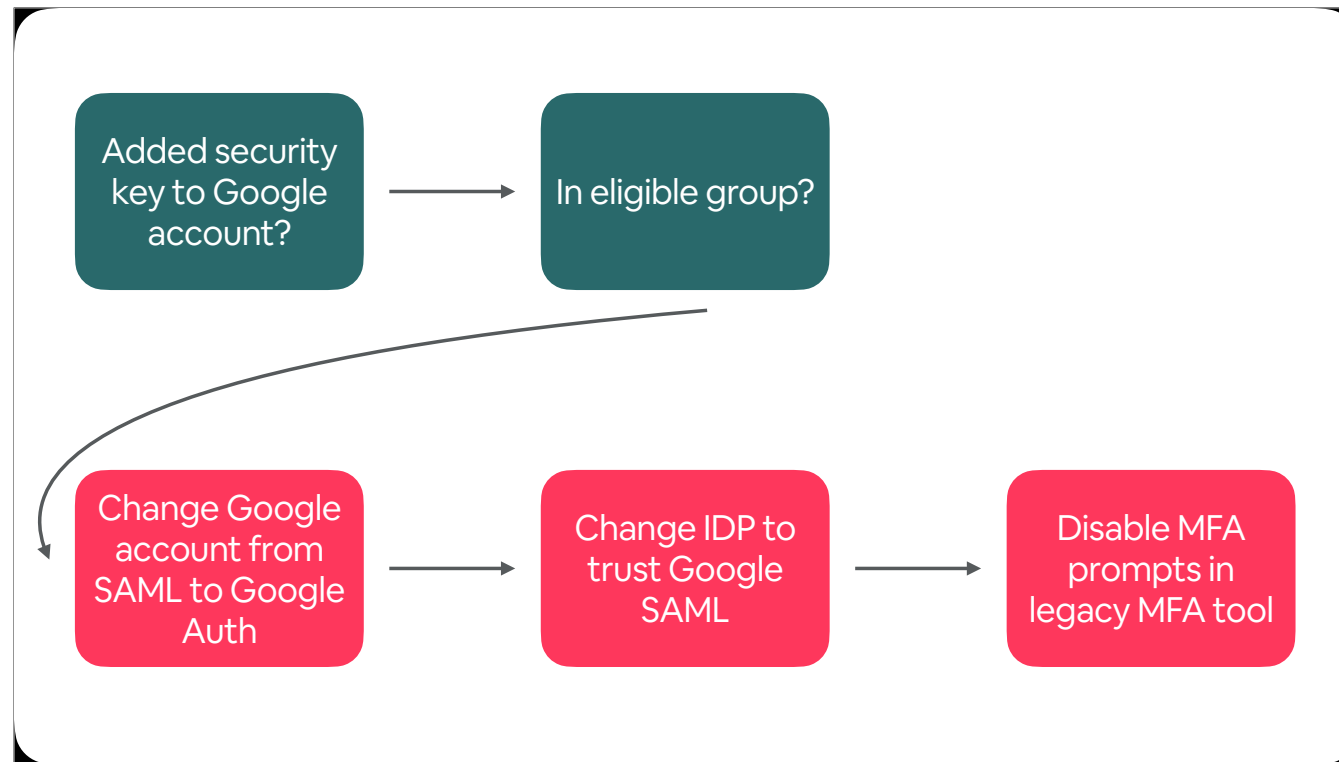
What will happen if we do not do this? This is where you can outline the risks associated with not doing this. For us, it was “When Google releases Chrome 98, users will not be able to log into their devices”

When is this happening? As accurate of a timeline as possible. Set a deadline. Build slack into it - there will always be stragglers, some reason they left it to the last minute and found an issue - whatever - just make sure you give yourself chance to get everything done

Getting the communication lines open early in the project was critical - even if the timeline is still unclear, you can begin talking to your stakeholders to get this onto their radar.

How are we actually going to do this?

So how are we actually going to get this done?

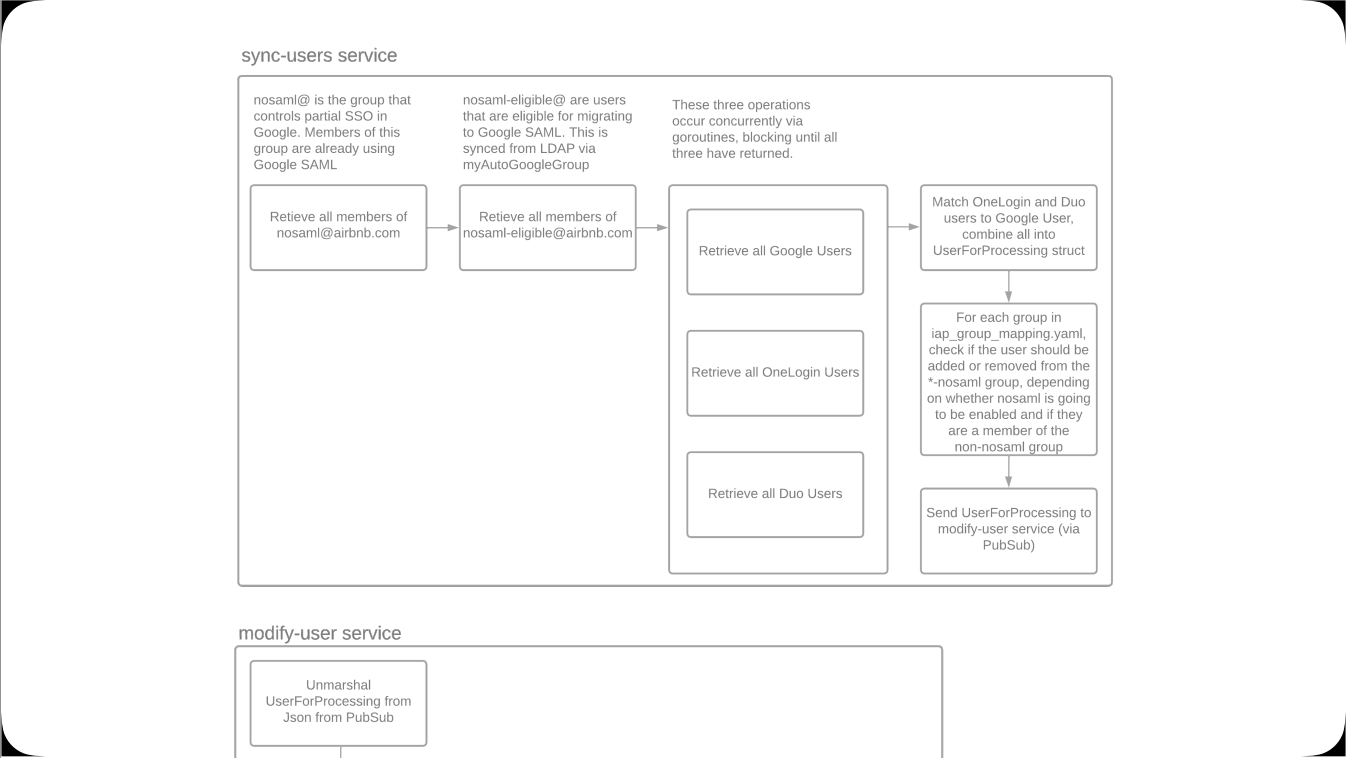


Our plan was to use the presence of a security key on the user's google account, plus whether they were in an "eligible" group to make the switch from using our actual IDP to using Google for their primary authentication, setting our IDP to trust Google's SAML assertion for that user and disabling the MFA prompt from the MFA provider

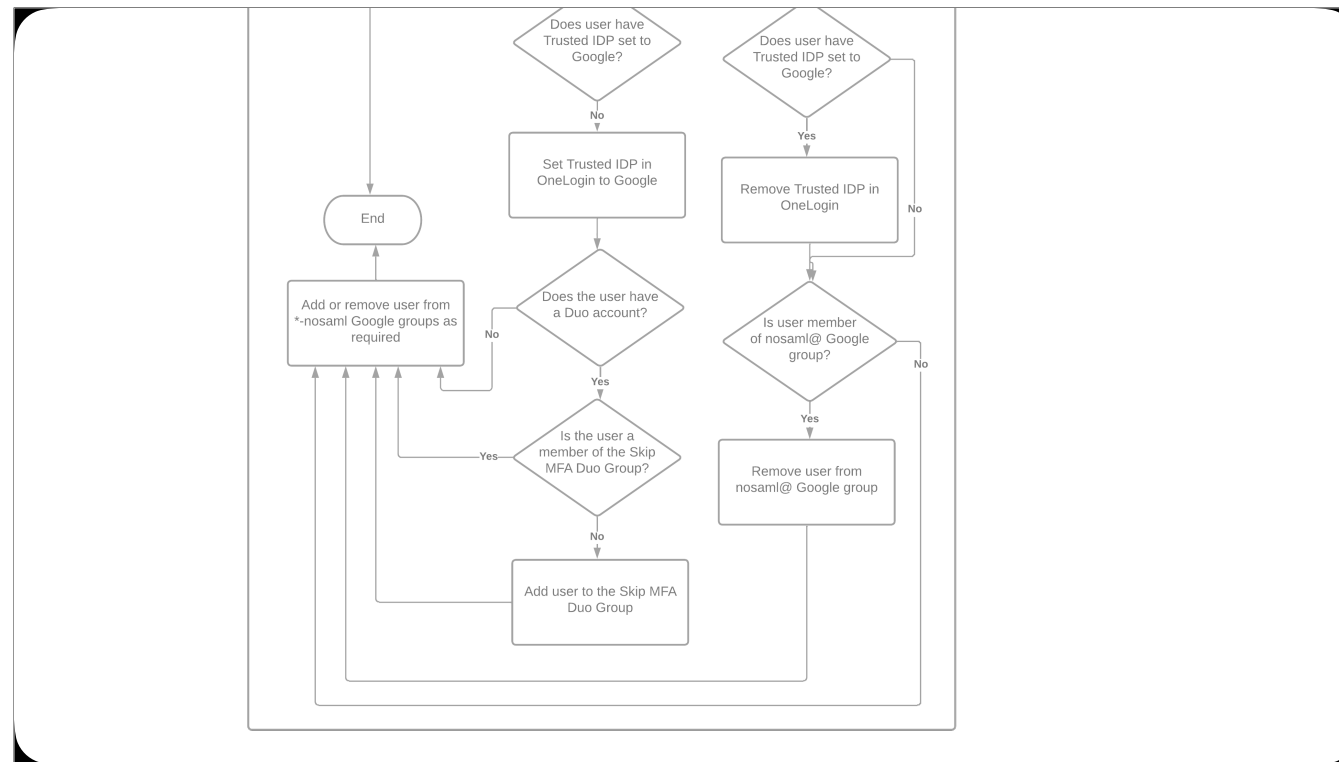


Sounds simple, right?

Yeah, not so much. There were lots of things to consider here - how do we handle rolling this back if we needed to (which thankfully we didn't), how do we handle exceptions if someone has a strange app that they need more time to migrate over to modern mfa?



{CLICK TO START ANIMATION}



In the end this was the diagram for our migration tool - I'm not going over every single step, but you can see it covered many eventualities. We also had quite a few challenges when developing it - some vendor's api documentation was plain wrong, some vendor's api's were much slower than the others and we also hit scaling limits, which ended up with us splitting the app into micro services so we could scale horizontally

Validation and verification

We obviously weren't just going to yolo this change out across our entire customer support user base.

Choosing the right testers is critical

We chose a CS partner that we have worked with before to test changes - we know we have a good relationship with them, they have good IT folks who can articulate issues quickly, and they are also in EMEA, which means we can test at the end of their shift so we don't cause them too much disruption. The time difference also means that we have time to roll back changes if they go badly before they start work the next day, and it gives us a chance to work on any issues during our day on the west coast of the US so we can get them testing again the next day



Make sure lines of communication are clear

We created a slack channel and made sure all of the stakeholders both on our side and the partner side, such as their IT folks, the team leads who would be talking directly with the agents, were invited. We made sure that all comms around testing stayed in there - it saved making sure the right people were on the email thread, and allowed people to catch up on what has already happened when they were added to the project

Issues caught in testing

- Confusion about how long the process would take
- Confusion about why they weren't prompted for the key immediately

During our tests with this partner, we found two main issues - both around our comms. As I mentioned before, some of the APIs we had to work with were really slow, it could take up to 90 minutes for a run of the migration process to complete - our users were surprised when they were still on the old login flow immediately after enrolling their security key. We also had users who expected to need to re-authenticate immediately - they already had a Google session, we didn't need to invalidate it.

Launch day

So our testing was complete, we were ready to launch. Our main task at this point was getting the message in front of our users. The actual actions they needed to take were very simple. We are fortunate to have great partners in our change management org - they sent repeated comms to our CS partners, kept lists of people up to date who hadn't migrated, and kept on top of the sites that weren't in compliance.



注意

请在2022年2月4日之前将你的安全密钥注册至谷歌



注意

请在2022年2月4日之前将你的安全密钥注册至谷歌



注意

请在2022年2月4日之前将你的安全密钥注册至谷歌



注意

请在2022年2月4日之前将你的安全密钥注册至谷歌

We also {click}



お知らせ

2022年2月4日までにGoogleに自身のセキュリティキーを登録してください。



お知らせ

2022年2月4日までにGoogleに自身のセキュリティキーを登録してください。



お知らせ

2022年2月4日までにGoogleに自身のセキュリティキーを登録してください。



お知らせ

2022年2月4日までにGoogleに自身のセキュリティキーを登録してください。

Had wallpapers {click}



알림

2022년 2월 4일까지 보안 키를 Google에 등록하세요.



알림

2022년 2월 4일까지 보안 키를 Google에 등록하세요.



알림

2022년 2월 4일까지 보안 키를 Google에 등록하세요.



알림

2022년 2월 4일까지 보안 키를 Google에 등록하세요.

Localized into {click}



Each of the major languages our agents spoke to remind them of what they had to do.

Enforcement

So eventually we needed to enforce our new security posture

Six weeks

Project start to enforcement

We gave our partners six weeks to migrate. You will never get to 100% compliance just by asking nicely, so we closely monitored the enrollment numbers. But of course, despite the hard deadline we couldn't shut down our entire contact center operations if no one had enrolled. When the time came, we had almost 95% compliance before the deadline - we decided to block access to tools the agents needed to use if they hadn't enrolled - this definitely encouraged them to enroll quickly



The end!

And that's the end of our story! {CLICK} For the first 60% of our users. The next 40% were much more interesting

The end!

(For 60% of users)

Webauthn for Airfam

We still needed to get our employees and contractors using security keys

Goals

No push notifications

No SMS

No one time passcodes (OTP)

Lets look at our goal for the migration again - to eliminate phishable mfa from our environment - this means no push notifications, no SMS, no one time passcodes

The motivation

Let's look at a real world example of why you want to do this. Last year a large ride sharing company was compromised. How did the attacker get in? First off a user's password was found on the dark web - so the attacker tried to log in to the user's account.

Push notification

Which meant that a push notification was sent to the user's phone. This happened {click}

Sent over

over

And over

And over

And over again

And over again

Until one was accepted

Until one was finally accepted

**Accepted in a completely
different location to the user**

So the actual MFA happened in a totally different place to where the login was happening. And as soon as the user accepted one push, the attacker was able to add their own mfa device to the user's account.

Push MFA is insecure

Lots of other bad things happened once the bad actor got past the perimeter and could get onto VPN, but the fact remains that push mfa is one of the primary reasons this company was compromised. So we planned on beginning the migration of the rest of our users at the end of 2022, after the summer travel peak was over

The world changed in February 2022

But the world changed a little bit in 2022. World events made us want to accelerate our migration

Getting buy-in

One of the biggest reasons push is popular, is because it is super convenient.

Security keys are inconvenient

And unfortunately security keys are inconvenient. We were able to allay some of these fears by only requiring users to mfa once every two weeks on managed devices in most cases and rely on Google's suspicious login detection

What happens if I lose my security key?

There was the very valid fear of not being able to work if users lose their key. We decided to give users two keys - one that lives in their laptop permanently and another that they can use either as a backup on their laptop and for mobile MFA using NFC (show blubikey). We also got buy in to make it so there is zero friction to ordering security keys - we decided that there didn't need to be any approvals, users were to be able to log into a self service portal to order their own security key

What is happening?

Why is it happening?

What will happen if we do not do this?

When is this happening?

So once again, here is how to get buy in for disruptive changes

- We are moving everyone from other, more easily phishable MFA to security keys
- We are doing this because we have security concerns with continuing our use of push mfa
- If we do not do this, we are concerned we will have a security breach, which could result in private information being compromised.

Planning

Applications

Security key ordering

Rollout plan

They were sold. Fortunately our migration code for CS partners could also be used for everyone else, but we did have other, logistical challenges.

The first, and one of our biggest challenges was to identify all of our applications that users could go through an SSO flow in. Obviously web apps that users were using a regular browser with were fine, but we had many native apps on both desktop and mobile that used ancient APIs that didn't support security keys.

We had to push our vendors a LOT to get these updated. We had a few apps that were potentially deal breakers for this migration - but we also found that some just needed to have a checkbox turned on to enable authentication via a real web browser rather than their web view, so remember to check in with the vendor and remind them what you pay for.

We needed a plan for getting security keys into our users hands - where were they shipped from, how do users order one, how long will they take to get delivered all over the world?

And finally, how are we going to enable this for our users? We had high priority users who we needed migrating as soon as possible - our execs and their direct reports. We didn't have a self service method to get them security keys yet, so our awesome support folks worked with all these users to get them keys - in some cases even hand delivering them.

Next we wanted our IT and InfoSec people to get on board early to catch any potential issues, then we could start asking the rest of our users to enroll. We decided to not prevent anyone from enrolling at any point - if they heard about it and wanted to try it early, we were definitely thankful

for more testers.



Do all the things

Once this part of the project was rolling, we met twice weekly to ensure everyone stayed aligned - we had an awesome PM keeping us moving in the right direction, and we identified our minimum product early on - we decided which apps were critical to the business and would block the launch and those which we could live without - for example a native app that has the same functionality in it's web version.

Launch day (again)

The day finally came - we had the self service portal to order keys ready, the automation was ready to move people to the new authentication flow when they enrolled their security keys. And some people in IT even ordered and enrolled keys. But a lot did not. We really needed these people to move onto the new flow to help iron out any issues before we launched to the whole company. This is where we made our first really big mistake

Who makes the call?

We knew we would need to enforce the use of security keys on our testers at some point. But whilst we had buy in on the project from senior leaders, we hadn't solidified who would sign off on cutting off access to a whole bunch of people. Fortunately, our leadership really believed in the project, so our CIO posted in slack a few times encouraging people to enroll (with the all important why). When it came to deadline day, our director didn't hesitate to give us the final stamp to block people out.

Identify decision makers

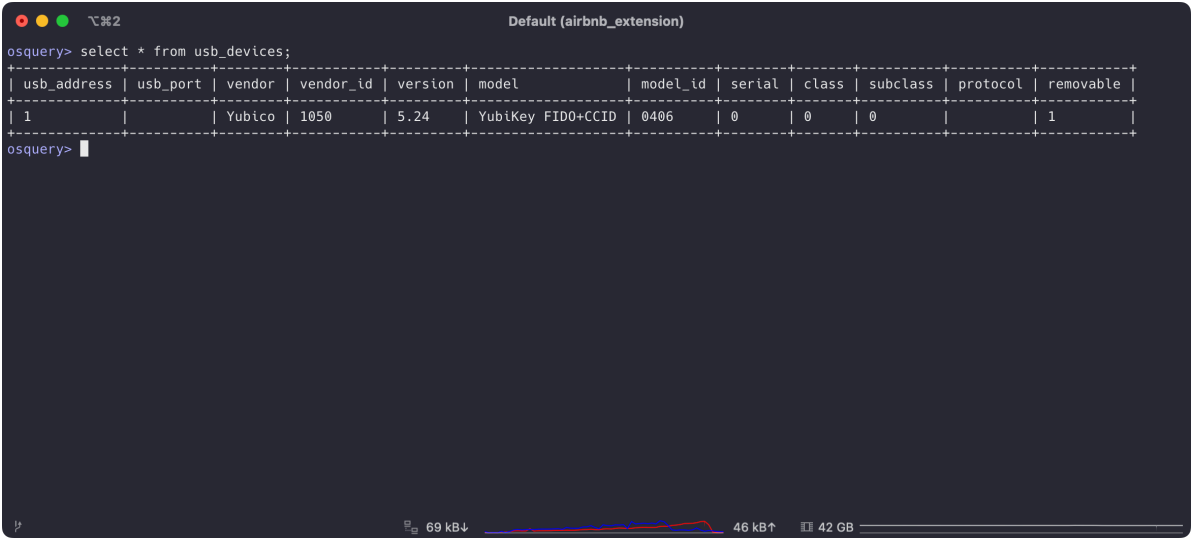
The moral of this story is to identify those who can ratify the hard decisions early on, and make sure you prep them about the decision they will be making.

Enforcement: IT & InfoSec

So it was time to actually enforce for our first group

How did we know we could enforce?

Whilst we knew the laptops had security keys in when we shipped them, we didn't know if they had been lost, eaten by the dog, whatever



```
osquery> select * from usb_devices;
```

usb_address	usb_port	vendor	vendor_id	version	model	model_id	serial	class	subclass	protocol	removable
1		Yubico	1050	5.24	YubiKey FIDO+CCID	0406	0	0	0		1

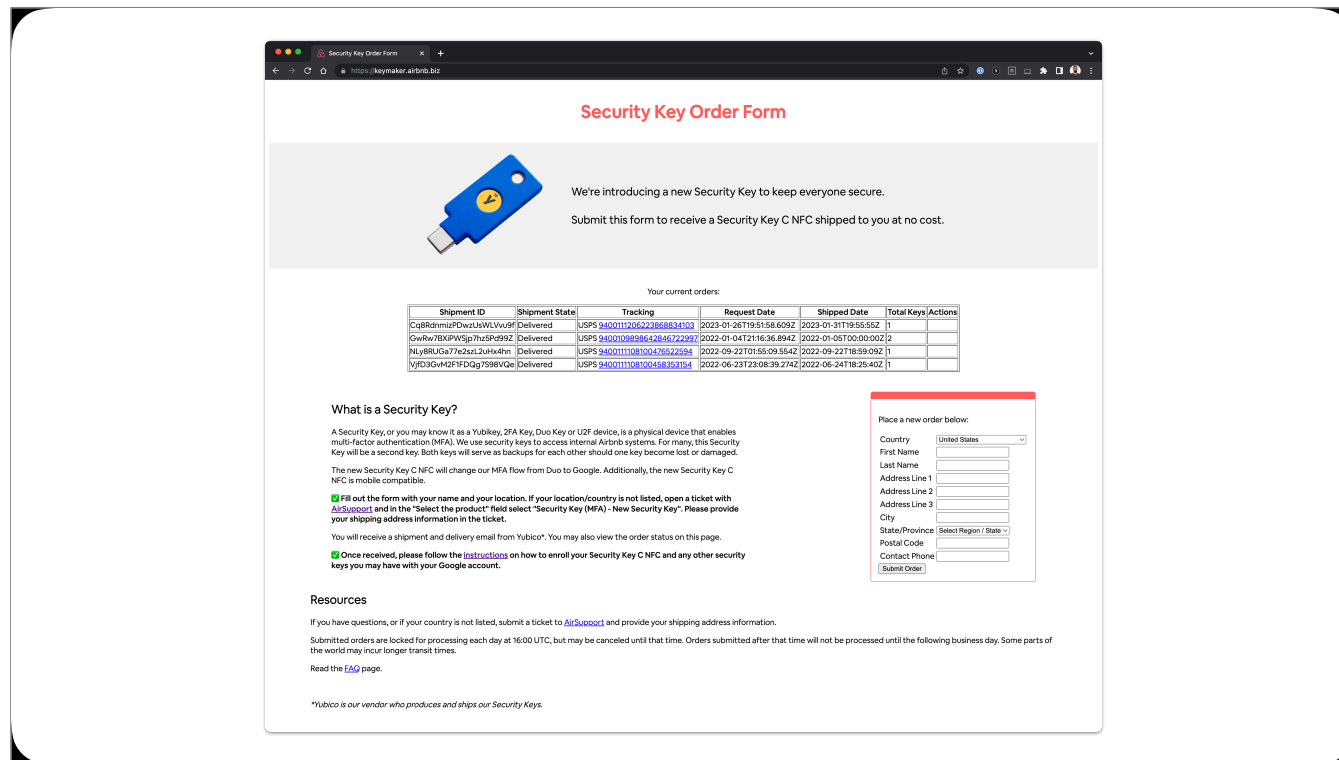
osquery>

69 kB↓ 46 kB↑ 42 GB

So how could we make sure? We have osquery streaming data about our endpoints into centralized logging - we could query each device and see which ones didn't have any security keys. We were able to add them to exemption groups whilst they got another key from our support folks. So we felt comfortable blocking access to some things like jira for those people that hadn't enrolled, and just like magic our enrollment numbers went up

How did enrollment go?

So apart from one application that we needed to add five or so users from IT to the exemption group for whilst they shouted at their vendor, the rollout was going pretty smoothly. So lets take a look at the user journey



I mentioned before that we made it trivial for users to get a security key - this is the portal that we built. It integrates directly with our vendor's API so they handle all of the shipping for us. You can see that I ordered a few keys for, ummm, science? The important thing to take away here isn't that you need to build some custom portal to ship keys to people - it's that you need to make it as frictionless as possible for your org. If you are serious about everyone always having a security key, you need to remove all barriers to getting one, which includes approvals, and if possible any human interaction at all. In our tool they enter their address and in a couple of days they get a key delivered to their front door

11,000

Security keys shipped in 2022 to Airfam

All in all, we shipped 11,000 security keys in 2022 to our employees and contractors across the globe

Always be selling

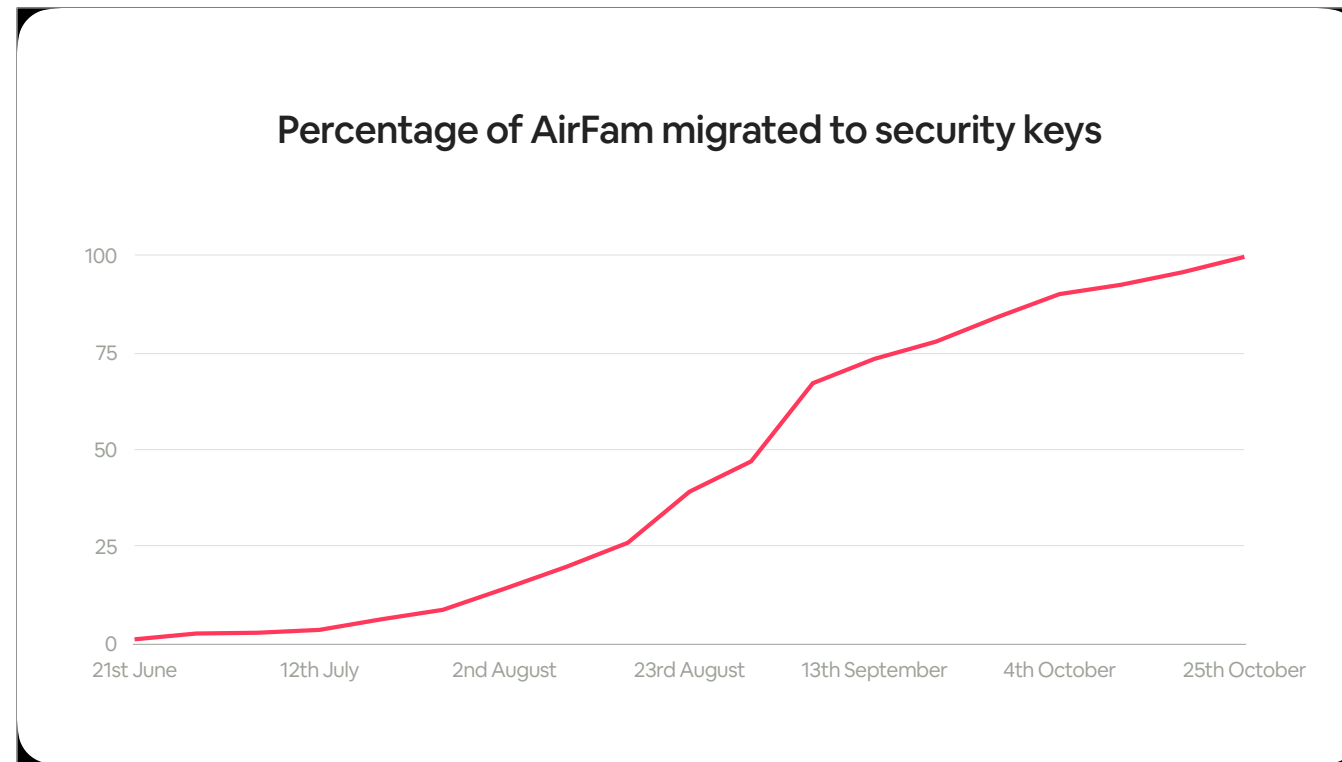
Weekly emails for unenrolled folks

Company wide Slack posts

Company newsletter

All hands meeting

At this point our marketing plan was in full swing. We were sending emails every week to people that hadn't enrolled, we had messages in the all company slack channel and we were in the monthly newsletter. We also got the chance to be featured before an all hands meeting with our media production team making a slightly tongue in cheek video in the style of those silly unboxing videos you see on YouTube



And this is how things went for a few weeks - every week we would email a new group and send reminders to those who had already had their first email. Our numbers increased pretty steadily, and we also started pinging the managers of people who hadn't migrated - after all, it's in their interests that their employees are able to access resources needed to do their job.

Enforcement

Eventually we got to our enforcement date. We had learned from last time and got our execs to sign off on it. We got up to about 97% compliance of active users when we blocked access to tools behind our identity aware proxy - the same enforcement method we used for our contact center partners - but a surprising number of people didn't log into things behind IAP so could avoid our enforcement method.

So in the end we just force migrated them to Google authentication. We started on the people who had most recently signed into their account, and moved them over, which effectively locked them out of their account. They had to get in touch with our support people, who made them go through identity verification to get back into their account. We performed this slowly over a few days as we knew that the only path these people had to get back into their account was to call our support folks, and we wanted to limit the load on them.

27,900

Users migrated in 8 months

In the space of eight months, we migrated nearly 28 thousand users who were located in almost every country on earth

Once again, this would only have been possible with the complete buy-in from our leadership, so if you only take one thing away from this talk its this:

Get buy in from the very top

Getting buy in not only gets you the resources you need - and it definitely takes quite a lot for a project like this - a total of five teams were involved across InfoSec and IT, and somewhere close to 70 different people involved in this migration - it was a huge effort and took a ton of work. But leadership buy in also gets you the political clout to make the tough decisions - you will invariably be blocking some portion of your users - how many can you block? How much backing will you get when you need to cut some people off? These are the things you get when you sell this to your leadership

Thank you!

<https://grahamgilbert.com>

<https://graham.at/movember>

[@grahamgilbert@mastodon.social](https://mstdn.social/@grahamgilbert)

graham.gilbert@airbnb.com

So that's it - you can find me on these places - I have a rarely updated website, I very occasionally post on mastodon and you can email me at my very cryptic email address. And if you found this talk useful in any way, please consider donating to the Movember foundation. And now we can take some questions