FREE DOWNLOAD

FIELD GUIDE / V1.0
2026 EDITION

IDENTITY & SECURITY FIELD GUIDE

Vertex Frontier.

The Entra ID Passkey Deployment Playbook

A practical field guide for IT admins before September 2026.

Microsoft is auto-enrolling every SMS and voice MFA user into passkeys starting September 1, 2026. This playbook walks you through the five mistakes that quietly undermine a passkey rollout, a 9-point readiness checklist, a PowerShell audit script, and the decision frameworks you need to enforce phishing-resistant MFA without locking users out.

AUDIENCE

Entra ID Administrators · Identity Leads · Security Architects

01 Executive Summary

WHY THIS GUIDE EXISTS

On **September 1, 2026**, Microsoft will start auto-enrolling every user currently registered for SMS or voice MFA into passkeys inside Entra ID. The telecom-based MFA methods organisations have relied on for years are not disappearing overnight, but the default is changing under your feet — whether your tenant is ready or not.

This playbook exists because **"enabled" is not "enforced,"** and a lot of IT teams are about to learn that the hard way. Flipping the passkey toggle in the Entra admin centre gives users the option to register a phishing-resistant credential. It does nothing to stop an attacker from walking straight past that option and using a weaker, already-registered method. The gap between "we have passkeys" and "we are actually phishing-resistant" is where most rollouts quietly stall — and where this guide focuses its attention.

It is written for the people who will actually do the work: **Entra ID administrators, identity leads, and security architects** planning a sequenced migration away from telecom-based MFA. It condenses a longer technical analysis into the most actionable pieces: the five mistakes that quietly undermine a passkey rollout, a 9-point pre-enforcement checklist, a PowerShell audit script, two decision frameworks (synced vs device-bound passkeys and provider/OS compatibility), a before/after comparison, three real-world case studies, and a quick-reference FAQ.

You can read it end-to-end in roughly twenty minutes, or jump directly to the checklist on page 4 and the script on page 5 if you are mid-rollout and need an answer now. Everything here assumes you have at least Entra ID P1 licensing for the Conditional Access pieces; registration and sign-in with passkeys themselves are available on every Entra ID edition, including Free.

> ***The one-sentence summary:*** *Passkeys stop time-of-issuance phishing, not post-issuance token theft. Plan enforcement, bootstrap, compatibility, and recovery before you flip the switch — not after the help desk lights up.*

02 The Five Mistakes at a Glance

FIELD-TESTED, REPEATEDLY PAINFUL

Each mistake below has been seen in production rollouts. They are listed in the order they typically bite: licensing and enforcement first, then bootstrap lockouts, then a misunderstanding about what passkeys actually protect against, then device compatibility, and finally the assumptions people make about Microsoft Authenticator. A condensed fix follows each one; the full decision frameworks are in later sections.

Mistake 1 — Assuming "Enabled" Means "Enforced"

Turning on the Passkey (FIDO2) policy in the authentication methods menu makes passkeys *available* to your users. It does not make them *preferred*, and it does not make them *enforced*. Those are three separate stages, and conflating them is where most rollouts stall. An adversary-in-the-middle (AiTM) phishing kit simply tells the browser it does not support passkeys; if your Conditional Access policy only demanded "MFA" rather than a specific authentication strength, Entra happily falls back to SMS or any weaker method the user still has registered.

- **The fix:** Turn on system-preferred authentication (or leave it Microsoft-managed), then build a Conditional Access policy using **Require authentication strength → Phishing-resistant MFA** — not the generic "Require multifactor authentication" control. This needs Entra ID P1. If you want finer control, build a custom authentication strength scoped to specific methods, down to a passkey provider by AAGUID.

Mistake 2 — Creating a Bootstrap Problem You Did Not See Coming

If you require a passkey to sign in but a separate Conditional Access policy requires MFA before a user can register a new authentication method, you have built a deadlock. The user needs MFA to register the passkey that would have satisfied MFA. This is a genuinely common way for enforcement rollouts to turn into a support queue overnight, and it is also the cleanest gap to close if you plan it in advance.

- **The fix:** Run a registration campaign for at least two to three weeks before enforcement goes live, and configure **Temporary Access Pass (TAP)** with role-based restrictions as your bootstrap and recovery path. A TAP is a time-limited passcode that satisfies MFA on its own — making it the cleanest first credential for new joiners and the cleanest recovery path for users who have lost a device.

Mistake 3 — Assuming Passkeys Stop All Token Theft

Passkeys are phishing-resistant. They are not theft-proof. The distinction matters: passkeys defend against **time-of-issuance theft**, where an attacker sits between the user and Entra during sign-in and steals the session token as it is handed out. They do nothing against **post-issuance theft**, where infostealer malware lifts the session token straight off the endpoint after a legitimate, passkey-secured sign-in has already completed. Microsoft's 2025 Digital Defense Report attributes roughly 80% of recent MFA-bypass breaches to session-token theft, typically via AiTM kits.

- **The fix:** Treat passkeys as one control in a stack, not a replacement for endpoint security. Keep EDR and application control genuinely effective on every device that can reach Entra-protected resources, and where your licensing supports it, layer in token protection so a stolen session token cannot be replayed from a different device. Phishing-resistant MFA closes the front door; it was never supposed to be your only lock.

Mistake 4 — Not Checking Device Compatibility Before You Enforce

Passkey support is not a single yes/no answer. It depends on the OS version, the browser or app in play, the passkey provider, and whether sign-in is direct on the device or cross-device via QR code and Bluetooth. Test exclusively with passkeys stored in Microsoft Authenticator and you will conclude your users need Android 14 or iOS 17 or later; allow synced passkeys through Google Password Manager instead and the floor drops to Android 9. That is a several-year shift in your realistic minimum OS, driven entirely by which provider you standardise on.

- **The fix:** Inventory registered device OS versions and browsers through Entra sign-in logs and Intune before you set an enforcement date. Decide deliberately whether you are permitting synced passkeys, device-bound passkeys in Authenticator, or both. Also check whether any Intune or GPO policy blocks Bluetooth outright — a legacy DLP decision that will silently break cross-device authentication.

Mistake 5 — Assuming Authenticator Syncs Passkeys Like It Syncs Everything Else

A passkey registered in Microsoft Authenticator is device-bound. It does not move with an app backup or a new-device restore the way other Authenticator data does. Lose the phone, and you have lost that passkey — full stop. That is a deliberate trade-off: device-bound credentials offer stronger attestation guarantees, which is exactly why Microsoft's own guidance points admins and highly privileged users toward device-bound options rather than synced ones. Synced passkeys (in iCloud Keychain or Google Password Manager) follow the user across devices automatically but with weaker attestation.

- **The fix:** Set expectations up front that Authenticator passkeys do not survive a device change, and build device-loss recovery around Temporary Access Pass rather than an ad-hoc support-desk workaround. Then make a conscious, documented decision about which user segments get synced passkeys and which get device-bound ones — do not let the default apply uniformly by accident. Microsoft's own consumer data shows synced passkeys hit a 99% successful registration rate and sign users in roughly 14× faster than a password-and-MFA combination.

03 Pre-Enforcement Readiness Checklist

PRINT THIS PAGE. TICK EVERY BOX BEFORE YOU SET A DATE.

Nine checks, pulled directly from the five mistakes above. Run through them once before you set an enforcement date — not after the support tickets start. Each item maps to a specific configuration in your tenant and is binary: it is either done or it is not. Print this page and tick the boxes, or paste it into your rollout tracker as the gating criteria for moving from pilot to enforcement.

Done	Mistake	What must be true before you enforce
☐	**M1** Enforcement	Conditional Access uses **"Require authentication strength → Phishing-resistant MFA"**, not the generic "Require MFA" control. Verified in the policy Grant section.
☐	**M2** Bootstrap	A **registration campaign** has been running for at least two to three weeks before any enforcement policy goes live, with snooze counts configured.
☐	**M2** Recovery	**Temporary Access Pass (TAP)** is configured and role-restricted for onboarding new joiners and handling device-loss recovery scenarios.
☐	**M3** Endpoint security	**EDR and application control** are actively maintained on every device that reaches Entra-protected resources — not treated as solved by passkeys.
☐	**M4** Compatibility	Registered device **OS versions and browsers** have been pulled from Entra sign-in logs or Intune before setting an enforcement date.
☐	**M4** Bluetooth	Confirmed whether any **Intune or GPO policy blocks Bluetooth**, and scoped an exception for FIDO2 authenticators via the Bluetooth Policy CSP if it does.
☐	**M4** Remote access	Confirmed whether **RDP, Windows 365, or jump host tooling supports WebAuthn redirection** before enforcing on remote or contractor accounts.
☐	**M5** Passkey type	A conscious, documented decision has been made on which user segments get **synced passkeys vs device-bound passkeys** — not left on the Entra default.
☐	**M5** Recovery path	Users have been told, before it happens, that a **device-bound Authenticator passkey does not transfer to a new phone**. Recovery via TAP is documented.

Five mistakes, nine checks. Run through them once before you set an enforcement date — not after.

04 PowerShell Audit Script

FIND OUT WHO IS ACTUALLY READY BEFORE YOU ENFORCE

Before you set an enforcement date, you need to know who in your tenant is actually ready. Rather than clicking through each user's authentication methods in the admin centre one at a time, the Microsoft Graph PowerShell SDK exposes this directly. The script below pulls every user in the tenant and lists who has a registered passkey (FIDO2) method, alongside when it was created and which provider it came from — the exact list you need before scoping a Conditional Access policy to "ready" users first.

Run this against a pilot group before you touch anything tenant-wide. Cross-reference the output against your Conditional Access target group, and you have a factual answer to "who would this policy actually block right now" instead of a guess.

PowerShell (Microsoft Graph SDK)

```
# Requires: Microsoft.Graph.Authentication, Microsoft.Graph.Users, Microsoft.Graph.Identity.SignIns
# Scope needed : UserAuthenticationMethod.Read.All (application or delegated)
# Purpose : List every user with a registered FIDO2 passkey, with provider and attestation level.
# Run against a pilot group first; cross-reference against your Conditional Access target group.

Import-Module Microsoft.Graph.Authentication, Microsoft.Graph.Users, Microsoft.Graph.Identity.SignIns

Connect-MgGraph -Scopes "UserAuthenticationMethod.Read.All" -NoWelcome

Get-MgUser -All -Property Id, UserPrincipalName | ForEach-Object {
$user = $_

Get-MgUserAuthenticationFido2Method -UserId $user.Id | ForEach-Object {
[PSCustomObject][]
User = $user.UserPrincipalName
PasskeyName = $_.DisplayName
Model = $_.Model
AttestationLevel = $_.AttestationLevel
Created = $_.CreatedDateTime
}
}
| Format-Table -AutoSize

Disconnect-MgGraph
```

Direct Graph REST Endpoint (non-PowerShell)

If you would rather query Graph directly without the PowerShell SDK — for example, from a Logic App, a scheduled Azure Function, or a reporting pipeline — the same data is available at the endpoint below. Use it with application permissions (UserAuthenticationMethod.Read.All) and a client-credentials flow.

```
GET https://graph.microsoft.com/v1.0/users/{id}/authentication/fido2Methods
```

```
# Returns: id, displayName, createdDateTime, model, attestationLevel,
# aaGuid (identifies the passkey provider).
```

> ***Pro tip:*** *Export the script output to CSV and join it against your group membership for the planned Conditional Access target. The intersection is your "safe to enforce on" list; the difference is your "still needs a registration campaign nudge" list. This single check prevents most enforcement-day lockouts.*

05 Decision Framework: Synced vs Device-Bound Passkeys

THE SPLIT MODEL IS THE PRACTICAL CONSENSUS

This is the decision most admins get backwards: the security-conscious instinct to default everyone to device-bound passkeys is usually the wrong call. Device-bound credentials offer stronger attestation — Entra can verify the specific device that created the key — but at a materially higher support burden. Synced passkeys trade some attestation rigour for a dramatically better adoption rate and a quieter help desk. Microsoft's own consumer data, drawn from hundreds of millions of accounts, is unambiguous on the trade-off for the general workforce.

Dimension	Synced Passkeys	Device-Bound Passkeys
Storage location	Cloud passkey provider (iCloud Keychain, Google Password Manager)	On the specific device that created the key (e.g. Authenticator, hardware key)
Sync across devices	Yes — follows the user via their platform account	No — does not move with app backup or device restore
Attestation strength	Weaker — provider-managed key material	Stronger — Entra verifies the specific device
Recovery burden	Low — sign into platform account on new device	High — requires TAP or admin-assisted re-registration
Successful registration rate	**99%** (Microsoft consumer base data)	Lower — friction from device-specific setup
Sign-in speed vs password + MFA	**~14× faster** (~3 s vs ~69 s)	Comparable to synced when available
Success rate vs legacy auth	**95%** vs 30%	Comparable, when the device is in hand
Recommended for	**General workforce** — standard employees, contractors without privileged access	**Admins and privileged roles** — sensitive systems, standing access

> ***The dependency to acknowledge:*** *A synced passkey ties the security of a corporate account to the security of the user's personal Apple or Google account. If that personal account is compromised, the synced passkey potentially goes with it. That is a real dependency — which is exactly why device-bound passkeys stay the right choice for admins and privileged roles, even as the general workforce moves to synced.*

06 Provider & OS Compatibility Matrix

PROVIDER CHOICE SHIFTS YOUR MINIMUM OS BY YEARS, NOT ROUNDING ERRORS

Passkey support is not a single yes/no answer; it depends on the operating system version, the browser or app in play, and whether you are authenticating directly on the device or across devices via QR code and Bluetooth. The table below summarises the practical floors for each passkey provider in common enterprise use. Always cross-check this against your actual device inventory from Entra sign-in logs or Intune before you set a compatibility bar for your whole organisation.

Provider	Platform	Minimum Version	Notes
Microsoft Authenticator	Android	Android 14+	Restrictive; narrows your fleet substantially.
Microsoft Authenticator	iOS	iOS 17+	Same restriction applies on iOS.
Google Password Manager	Android	Android 9+ (API 28)	**Widest Android coverage** if synced passkeys are allowed.
iCloud Keychain	iOS / macOS	iOS 16+ / macOS Ventura+	Synced passkeys across Apple devices.
Third-party password manager	Android	Android 14+	Bitwarden, 1Password — conflicts with Authenticator default.
Cross-device sign-in (QR + BLE)	Desktop → phone	Bluetooth enabled	Check Intune / GPO Bluetooth block first — see below.
Windows 10 (platform passkey)	Windows	Works, less reliable	QR prompts fail to appear; weaker WHfB integration.
Windows 11 (platform passkey)	Windows	Windows 11 22H2+	Recommended for managed Windows fleets.

Pro tip: Check Your Bluetooth Policy First

Cross-device sign-in, where someone signs into a desktop using a passkey stored on their phone, works by scanning a QR code and then confirming physical proximity over Bluetooth. A lot of enterprises already have an Intune device restriction policy or a legacy GPO that blocks Bluetooth outright — a perfectly sensible data-loss-prevention decision made years before passkeys were on anyone's radar — and it will silently break cross-device authentication for exactly the users who need it most.

The failure is quiet: users see a generic "Passkey not found" or "something went wrong" message with no indication that Bluetooth is the cause. Microsoft supports scoping Bluetooth access to allow pairing exclusively with passkey-enabled FIDO2 authenticators via the Bluetooth Policy CSP, rather than opening Bluetooth back up entirely. If your compatibility testing only covers devices with their Bluetooth policy left at default, you will not catch this until a real user with a locked-down laptop tries it and fails.

07 Before vs After: What Changes When You Do This Properly

SAME FEATURES, SEQUENCED CORRECTLY

The gap between a rushed rollout and a sequenced one is not more effort — it is the same handful of Entra features, in the right order, instead of enabling one setting and hoping. The table below makes the difference concrete. Use it as a target state: if your tenant looks like the left column today, the right column is what you are working toward before the September 2026 enforcement window arrives.

Dimension	Before (rushed rollout)	After (sequenced rollout)
Conditional Access policy	Single policy requiring generic "MFA".	Requires a specific **authentication strength → phishing-resistant**.
MFA fallback	SMS still accepted as fallback → MFA downgrade attack possible.	No fallback to SMS; AiTM downgrade blocked by policy.
Registration campaign	None — users have no nudge to register before enforcement.	Ran for 2–3 weeks; bulk of users voluntarily enrolled.
Temporary Access Pass	Not configured; recovery is ad-hoc help-desk work.	Configured and role-restricted; handles onboarding + device loss.
Device inventory	Unknown — enforcement date set without OS / browser checks.	Pulled from Entra sign-in logs and Intune before any date was set.
Passkey type strategy	Default applied uniformly; no decision on synced vs device-bound.	Synced for general workforce; device-bound for admins / privileged roles.
Endpoint protection	Assumed solved by passkeys; EDR coverage uneven.	EDR and app control actively maintained; token protection layered on.
Result	False sense of security; actual exposure barely moves.	Genuine phishing-resistant baseline; quiet help desk.

08 Case Studies: Real-World Lessons

THREE INCIDENTS THAT SHAPED THE RECOMMENDATIONS IN THIS GUIDE

The recommendations in this playbook are not theoretical. They come from observed incidents where the choice of authentication method — or the absence of one — made a measurable difference to the outcome. The three case studies below illustrate the two threat categories passkeys are designed to address (and the one they are not).

Case 1 — AiTM Phishing Campaign (Microsoft-tracked, 2025)

The challenge. Microsoft's threat intelligence team tracked a reverse-proxy phishing operation across a single reporting quarter. The campaign could complete a real, working Microsoft sign-in — MFA included — before stealing the resulting session token. Victims did nothing obviously careless: they passed a CAPTCHA, signed into what looked exactly like a genuine Microsoft page, and completed MFA as trained.

The outcome. The campaign compromised **more than 35,000 users across 26 countries in three days**, part of roughly 8.3 billion email-based phishing threats Microsoft detected that quarter. The proxy sat between the user and Entra the entire time, relaying every step, then walked off with the token issued at the end.

The lesson. Training did not fail these users — the authentication method did. A phishing-resistant credential breaks this chain at the point of sign-in, because there is no shared secret or one-time code for the proxy to relay in the first place. That is the exact gap passkeys are built to close.

Case 2 — Cloudflare FIDO2 Keys Blocking Phishing (August 2022)

The challenge. Cloudflare's employees were hit by the same SMS-based phishing campaign that had just compromised Twilio — a convincing fake Okta login page, sent moments after a freshly registered look-alike domain went live.

The response already in place. Cloudflare did not rely on TOTP codes or push notifications for MFA. Every employee had been issued a FIDO2 hardware security key years earlier, required to access every internal application.

The outcome. Three employees fell for the phishing message and typed their username and password into the fake page. None of the three could get past the hard-key requirement, because the key's origin-binding meant it simply would not respond to a domain that was not the real one. Cloudflare confirmed no systems were compromised.

The lesson. Cloudflare has been explicit that this was not luck — it was the direct, designed-for consequence of choosing an origin-bound, phishing-resistant credential over a phishable one. It is also a useful reality check on user training: three experienced security-company employees still clicked the link and typed their password. The credential, not the click, was what held.

Case 3 — LummaC2 Infostealer Economy (Disrupted 2025)

The challenge. LummaC2, one of the most prolific infostealer families of the past two years, was harvesting saved credentials, browser cookies, and authentication tokens from infected Windows devices at industrial scale, packaging the results for sale on underground markets.

The outcome. Microsoft's Digital Crimes Unit coordinated with Europol, the FBI, and the DOJ to seize the malware's infrastructure. By the time the takedown happened, **more than 394,000 Windows devices worldwide had been infected in just a two-month window between March and May 2025**. The operation resurfaced within weeks under new infrastructure, because the malware-as-a-service business model is cheap to relaunch.

The lesson. Law-enforcement takedowns buy time, not immunity. None of this cares which MFA method you used — if an infostealer is sitting on the endpoint and your endpoint protection has not caught it, it can lift the session token straight out of the browser after a completely legitimate, passkey-secured sign-in. That is an endpoint security problem, not an MFA problem.

09 FAQ Quick Reference

THE SEVEN QUESTIONS THAT COME UP MOST OFTEN

Q1. Do I need a paid Entra ID licence to use passkeys at all?

No. Passkeys (FIDO2) are available in every Microsoft Entra ID edition, including Entra ID Free, so registration and sign-in do not require an extra licence. You only need **Entra ID P1** to *enforce* passkeys through Conditional Access (P2 is not required for this).

Q2. What actually happens to SMS and voice MFA in September 2026?

Starting **September 1, 2026**, users currently registered for SMS or voice MFA are auto-enabled for passkeys in the authentication methods policy. Microsoft is also moving telecom-based MFA to a paid, provider-based model through the Microsoft Security Store rather than including it by default, with further pricing and provider details due from September 18, 2026.

Q3. Are synced passkeys actually safe enough for a business, or should everyone use hardware keys?

For most users, yes — the adoption and success-rate data Microsoft has published from its own consumer base backs that up. Reserve device-bound passkeys and hardware FIDO2 keys for admins, privileged roles, and access to your most sensitive systems, where the added attestation assurance is worth the recovery overhead.

Q4. Can old Android phones use Entra passkeys?

It depends entirely on the provider. **Google Password Manager** supports passkeys back to Android 9 (API 28). Passkeys stored specifically in **Microsoft Authenticator** need Android 14 or later. Check your provider choice against your actual device inventory before setting a compatibility bar for your whole organisation.

Q5. What is the real difference between AiTM phishing and infostealer token theft?

AiTM happens during sign-in — a proxy sits between the user and Entra and steals the token as it is issued. Phishing-resistant methods like passkeys stop this because there is no shared secret for the proxy to relay. **Infostealer** malware steals a token *after* a legitimate sign-in has already completed, straight from the compromised device, and no MFA method — passkeys included — prevents that on its own. That is an endpoint security problem.

Q6. How do I avoid locking users out when I turn on Conditional Access enforcement for passkeys?

Run a registration campaign for several weeks first so most users have voluntarily enrolled, and set up Temporary Access Pass as the bootstrap and recovery path before you flip enforcement on. Test your Conditional Access policy in **report-only mode** and review who it would have blocked before making it live.

Q7. Does enabling passkeys automatically make them the default sign-in method?

No. Enabling the method, making it system-preferred, and enforcing it through Conditional Access are **three separate settings**. Skipping the middle two is the single most common reason a passkey rollout does not deliver the protection it was meant to.

10 Closing & About Vertex Frontier

PASSKEYS ARE ONE CONTROL IN A STACK

The recurring lesson across every rollout this guide is drawn from is the same: rushing a passkey deployment does not just risk a bad user experience — it risks giving you a false sense of security while your actual exposure barely moves. You end up with users who technically "have passkeys" and a Conditional Access policy that still lets an attacker walk in the side door.

The fix is sequencing. Run the registration campaign first. Configure Temporary Access Pass before you need it. Pull a real device inventory before you set an enforcement date. Make a deliberate decision about synced versus device-bound passkeys — and document it. And keep your endpoint protection genuinely effective, because passkeys solve phishing, not malware. None of that is more effort than a rushed rollout; it is the same handful of Entra features, in the right order.

If your Conditional Access setup is the piece holding your passkey rollout together — or the piece you are not confident about yet — that is the layer worth getting right before you touch enforcement. It is also the part of this stack most teams underinvest in relative to how much it is doing.

> ***One-line summary:*** *Passkeys stop time-of-issuance phishing, not post-issuance token theft. Plan enforcement, bootstrap, compatibility, and recovery before you flip the switch — not after the help desk lights up.*

About Vertex Frontier

Vertex Frontier is an identity and security consultancy focused on the practical side of Microsoft Entra ID — Conditional Access design, phishing-resistant MFA rollouts, privileged access, and the operational discipline that makes them stick. We write field guides like this one for the people who actually do the work: admins, identity leads, and security architects planning a sequenced migration rather than a single toggle.

If you found this playbook useful, the longer technical analysis it is drawn from — **"5 Microsoft Entra Passkey Mistakes That Quietly Undermine Your Security"** — walks through each mistake in more depth, with the underlying Microsoft guidance and the field notes behind every recommendation. Read it on the Vertex Frontier blog.