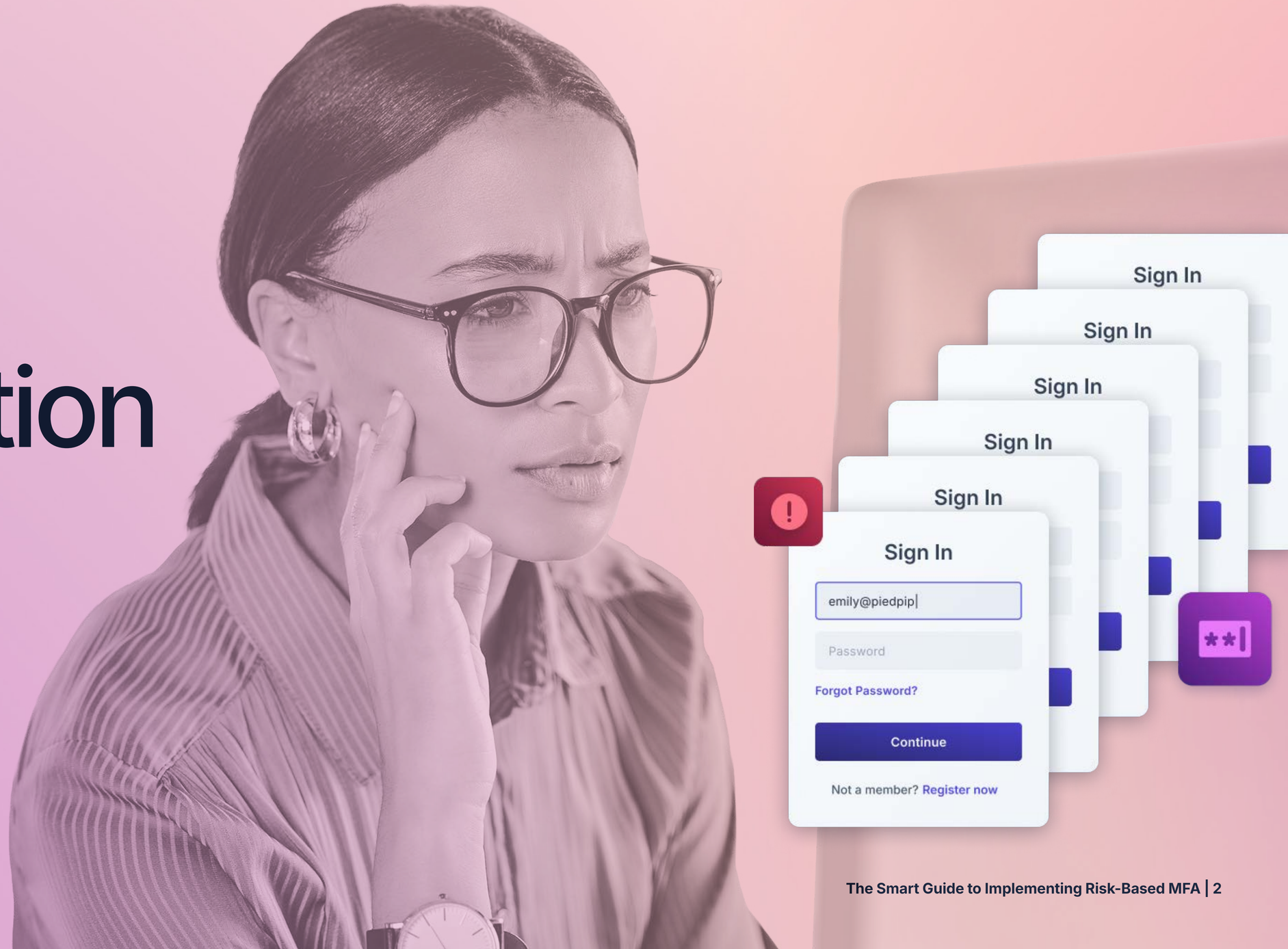
FusionAuth | Ebook

The Smart Guide to Implementing Risk-Based MFA That Customers Won't Hate

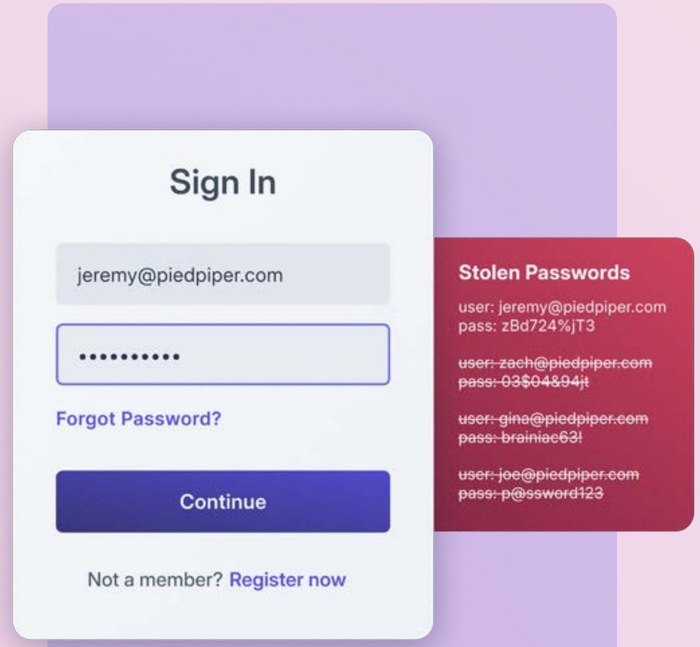
SECTION 01

Introduction



They Didn’t Hack In. They Logged In.

In 2023, attackers didn’t need stealthy tactics to wreak havoc on 23andMe, they just logged in.



Using passwords stolen from other breached sites, they spent five months going after 23andMe’s login page. Nothing flagged the traffic as suspicious. By the time anyone noticed, 14,000 accounts were gone[1].

The loss might sound contained. It was anything but. 23andMe’s DNA Relatives feature linked customers to biological matches across the platform. Those 14,000 accounts were connected to 6.9 million people’s genetic data. Ancestry, health predispositions, family trees, all exposed. Data customers had expected the company to protect.

Trust evaporated and it didn’t come back. Lawsuits hit hard and fast. So did account deletion requests.

In March 2025, 23andMe filed for bankruptcy. All because of some reused passwords and an authentication system that had no way to tell a legitimate login from an attacker working through a stolen credential list.

23andMe is an extreme case but the underlying problem isn’t. When authentication is too weak, attackers get in. If it’s too aggressive, customers walk out. Both cost you.

This guide walks through how risk-based Multi-factor Authentication (MFA) solves that problem, creating an authentication system that stops real threats without making legitimate customers feel like suspects.

[1] *Gene Petrino, “23andMe Data Breach: What Was Exposed, Who Was Affected, and What Happens to Your DNA Now,” Security.org, last modified July 8, 2026, https://www.security.org/identity-theft/breach/23andme/.*

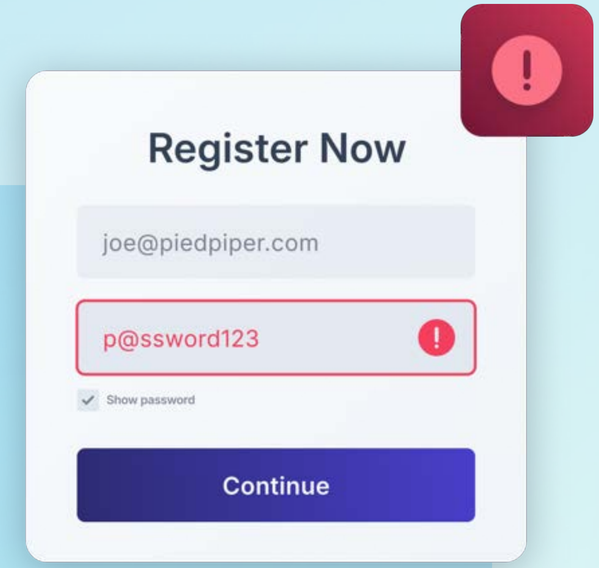
SECTION 02

Understanding the Landscape of Traditional MFA

Three Ways Attackers Defeat Traditional MFA

The weakness of passwords as an authentication mechanism is a well-known problem.

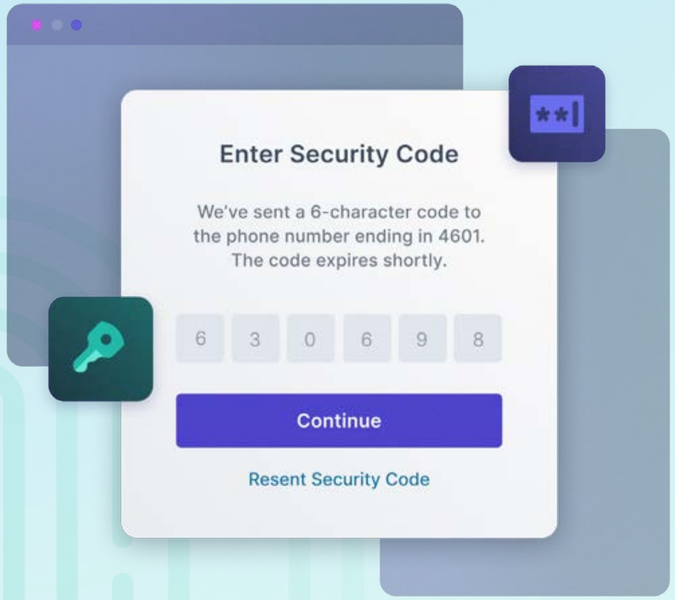
People reuse passwords across services. They choose predictable ones. They share them. And then, of course, they get stolen.



The 2025 Verizon Data Breach Investigations Report found stolen credentials behind 22% of breaches. That number has stayed stubbornly consistent year after year because the underlying economics keep working. Breached credentials don't expire. They get packaged, sold, and fed into an industrialized market where username and password combinations are tested programmatically against every service that will accept a login attempt. By the time your organization even knows a credential has been compromised elsewhere, it may already have been tried against your systems thousands of times.

Multi-factor authentication, a security method requiring two or more independent credentials to verify your identity, was the right answer to this problem. If attackers have your password but not your phone, they're stopped. For years, that logic held and MFA adoption climbed.

The problem is that attackers adapted in three specific ways that most MFA deployments still don't account for.



SIM SWAPPING

Attackers convince a mobile carrier to transfer a victim's phone number to a SIM they control. Once that happens, any SMS code sent to that number goes to the attacker instead. Carriers have tightened their processes, but attackers still pull it off regularly.

CODE PHISHING

Attackers call or message victims posing as IT support, a bank, or trusted service, and talk them into reading a code out loud or entering it on a page the attacker controls. The code is real but the request isn't.

AiTM PHISHING

In an adversary-in-the-middle (AiTM) attack, the attacker creates a fake login page that silently passes everything through to the real service. The user enters their credentials, completes MFA, and logs in successfully. They have no idea anything is wrong. But the attacker's fake site captured the logged-in session in the process and can now use it to access the real account, no credentials or MFA required. The user did everything right and still got compromised.

None of this means MFA is the wrong answer. It's still an essential component of an effective authentication strategy.

But the attack landscape is evolving every day. A system that challenges every login the same way, with the same method, regardless of context, treats all risk as equal. Attackers are aware of that weakness and they're taking advantage of it.

Can the MFA you have today protect your organization against the attacks that are coming for you tomorrow?

What Compliance Frameworks Require for MFA

Most standards require stronger authentication, but the details vary.

For a long time, compliance requirements around authentication left significant room for interpretation. That era is ending. Frameworks are now mandating specific capabilities that point toward stronger methods and more rigorous documentation requirements.

In Practice

Keep in mind that auditors don't just want to know that MFA exists. They want logs proving who was challenged, when, what method they used, and what the outcome was. This is where integration between your authentication platform and your SIEM or audit logging infrastructure matters. Every MFA challenge, every failure, every successful authentication should produce structured events your security team can query, report on, and export for auditors on demand.

The following chart details where the major frameworks stand.

Framework	MFA Requirement	What "Compliant" Looks Like
NIST 800-63B AAL2	MFA required	TOTP, push, or phishing-resistant method
NIST 800-63B AAL3	Phishing-resistant MFA required	Hardware keys or passkeys; non-exportable keys
PCI DSS 4.0	MFA for all CDE access (mandatory since March 31, 2025)	All access covered; no single-factor paths; logged events
HIPAA	No explicit mandate; risk analysis typically requires it	Risk analysis documentation; MFA for PHI systems
EO 14028	Phishing-resistant MFA for federal agencies	Passkeys or hardware security keys
SOC 2 Type II	MFA increasingly required for production access	MFA enforced on all production systems; evidence of compliance

Understanding Traditional MFA Options

Not all MFA methods provide the same level of protection.

Receiving a six-digit code by text message isn't the same as authenticating with a hardware security key that uses phishing-resistant cryptography to verify the user. Treating them as equivalent is a policy decision that often looks obvious only after an incident.

So which method should you choose and why?

Before comparing the available options, it's helpful to understand the four traditional authentication factors they build on.

1. SOMETHING YOU KNOW (PASSWORDS, PINS)
Anything stored in memory, like passwords, PINs, or security questions. It's the oldest factor and the weakest. Knowledge can be guessed, phished, stolen, or reused across sites. The 23andMe attack relied entirely on this category.

3. SOMETHING YOU HAVE (PHONES, HARDWARE KEYS)
A physical object (phone, security key, smart card) that can't be stolen remotely. Text codes are weak. Phone numbers can be SIM swapped. Hardware keys like YubiKeys are stronger, using cryptographic proofs that can't be phished or intercepted.

2. SOMETHING YOU ARE (BIOMETRICS)
Biometrics, typically your fingerprint, face ID, or voice recognition. The appeal is obvious: you can't forget your face and attackers can't steal it over the internet. The limitation is equally obvious: if biometric data is compromised, you can't reset it the way you'd reset a password.

4. SOMEWHERE YOU ARE (LOCATION)
Uses location as a signal, either your IP address, GPS coordinates, or network. It's the weakest standalone factor because location is easy to spoof with a VPN, and legitimate users travel. It's most useful as a risk signal ("this login is coming from a country this user has never been in") rather than a hard gate.

For most consumer applications, the practical question is which software-based factors to deploy across a user population that ranges from highly technical to barely tolerant of friction.

Four factors dominate real-world deployments.

1. SMS ONE-TIME PASSWORDS

While SMS earns some of the bad press it's gotten (SIM swap attacks are real), it offers a big benefit when it comes to customer authentication: completion rates. SMS is the factor users are most likely to finish without abandoning signup. The key is to stop relying on SMS the same way for every login. Risk-based MFA, which is discussed later, lets you decide when SMS is sufficient or when a stronger challenge is warranted.

2. EMAIL ONE-TIME PASSWORDS (OTP)

This method fits naturally into customer flows because the email address has already been collected, no extra setup or new device needed. The tradeoff? The second factor is only as strong as the user's email account. For low-risk actions or account recovery, email OTP is a reasonable option.

3. VOICE ONE-TIME PASSWORDS

Voice delivery works similarly to SMS. A one-time code is generated and sent through a configured messenger, with the code read aloud in a phone call. The practical case for voice is accessibility, for users who can't reliably receive or read SMS. There are some drawbacks. Because the code travels over the public phone network, SIM-swap and call-forwarding attacks are possible, but offering voice alongside SMS meaningfully improves completion rates without adding enrollment friction.

4. TIME-BASED ONE-TIME PASSWORDS (TOTP)

Rotating codes from Google Authenticator, Authy, or Microsoft Authenticator is the strongest option. Because codes are generated locally on the device and never transmitted over a carrier network, TOTP is significantly harder to phish at scale. Requiring this method can be a hard sell for the general consumer population, but it's a great option for customers who want stronger security and are willing to set it up.

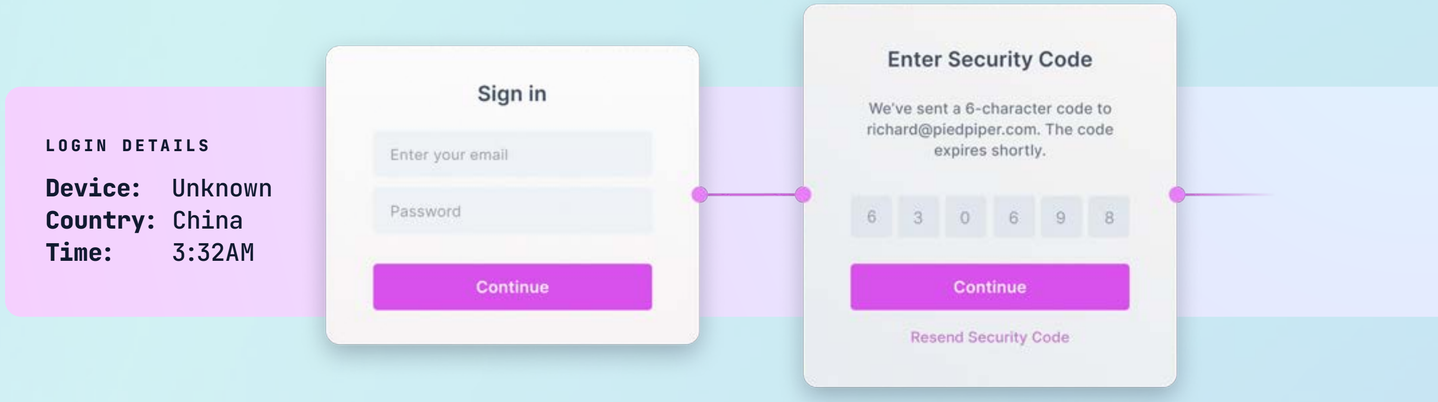
Why Static MFA Fails

Not all MFA methods provide the same level of protection.

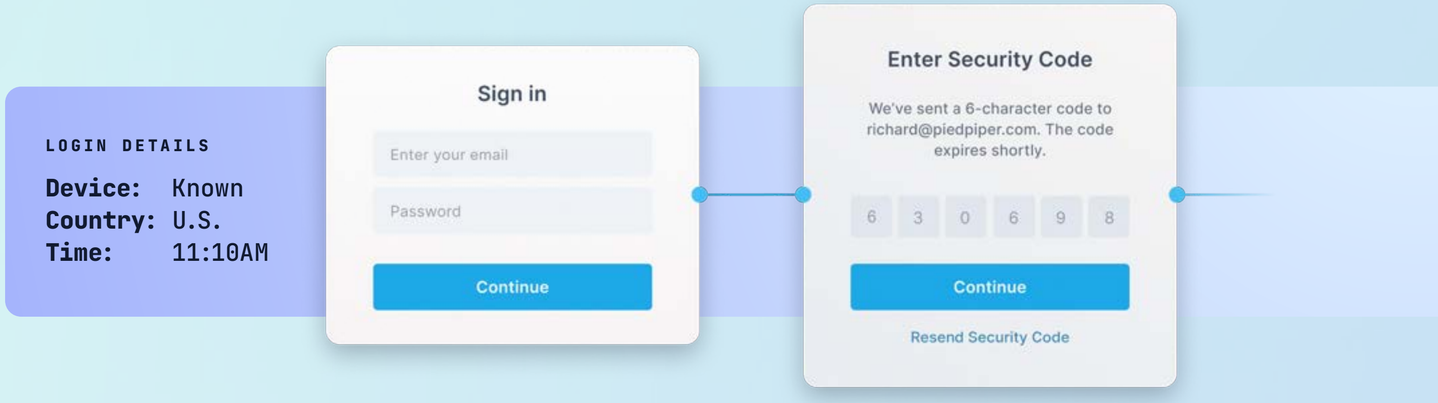
Here's a scenario that plays out millions of times a day: a person opens their banking app from the same phone, on the same home WiFi, at the same time they always check their balance and they get prompted for MFA. Everything about this login looks exactly like the 500 logins before it, but the system challenges them anyway. Meanwhile, that same account gets accessed from a device in another country at 3 a.m. and the system issues exactly the same challenge.

Same friction. Vastly different risk. This is the core problem with static, blanket MFA. It treats all logins as equally uncertain, over-challenging low-risk users while not challenging genuinely high-risk events enough. This is the problem that intelligent, risk-based MFA sets out to fix.

Abnormal login, unusual device, different country, odd time. Prompted for MFA.



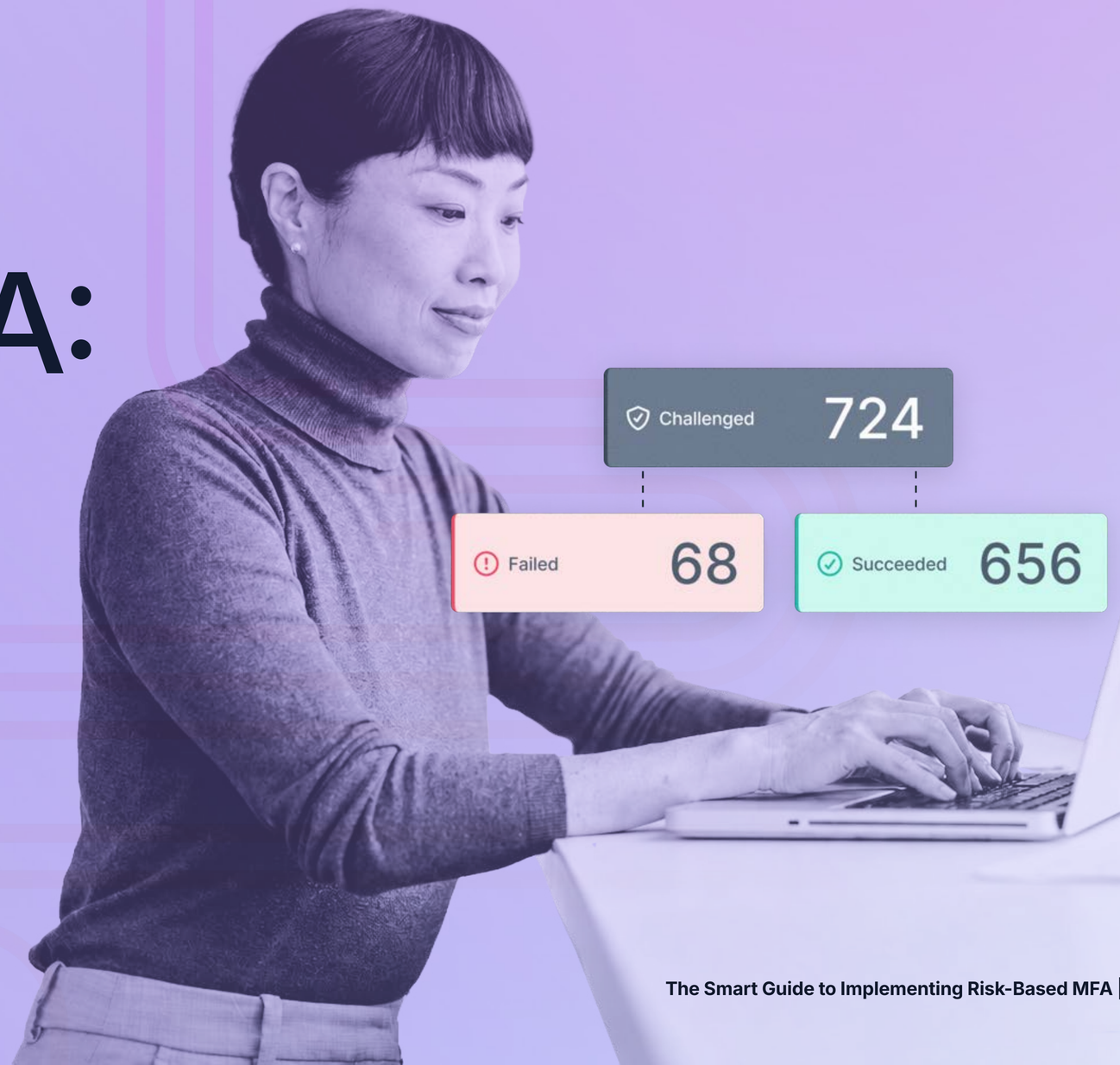
Standard login, usual device, usual location, usual WiFi. *Still prompted for MFA.*



SECTION 03

Risk-Based MFA:

What It Is, Why It Matters, and How to Get Started



What It Is and How It Works:

Risk-Based MFA

Risk-based MFA is an authentication approach that dynamically decides whether to challenge a user for a second factor based on the risk level of a specific login attempt, rather than challenging every user every time.

Instead of treating all logins the same, risk-based MFA evaluates each login against a set of risk signals and assigns a composite risk score of LOW, MEDIUM, or HIGH. The MFA challenge is only triggered when the risk score meets or exceeds a configured threshold.

Top 10 Risk Signals to Consider in Consumer Applications

A well-designed risk engine evaluates each login across multiple dimensions before a challenge decision is made. Identity vendors tend to approach risk engines in different ways, tracking different signals and using different algorithms to determine risk. However, there are 10 top signals your identity vendor should consider when triggering risk:

Risk Signal	Behavior Identifier
Impossible Travel	The account just logged in from a location physically impossible to reach since the last login, which strongly suggests credentials are being used by two different parties. One of them could be an attacker.
Untrusted Device	The user never marked this device as “remembered” during a prior MFA prompt, so there’s no standing signal that they personally vouch for it. Absence of that trust marker warrants re-verifying identity.
Unrecognized Device	This device has never been seen on the account before. New hardware is a classic indicator of credential theft or session hijacking, so confirm the person is really who they say they are.
Blacklisted IP	The login originates from an IP on a known-malicious set tied to attacks, botnets, or abuse. Traffic from these sources is far more likely to be hostile.
Stale Credentials	The password is many months or even years old, giving it a long window to have been leaked in a breach or guessed. Stale credentials are more likely already compromised.
Suspicious Password Reset	The password was changed recently, which is exactly what an attacker does right after taking over an account to lock out the real owner. A fresh change so recently is suspicious until proven legitimate.
Account Owner Change	A login identifier was added or changed recently, a common takeover move to hijack recovery and notifications. The recency of the change is the red flag, and risk drops the longer ago it happened.
Dormant Account Reactivation	The account has been dormant for a long time, a prime target since the real owner isn’t watching for unauthorized access. Reactivation after long silence often signals takeover.
Suspicious Browser	The browser/client string matches a list of known-bad user agents associated with attack tooling. Such fingerprints rarely belong to genuine human sessions.
Bot Detected	Input timing indicates a bot or automated system rather than a human. Automation at the login screen points to credential stuffing or scripted abuse.

None of these signals is definitive on its own. What makes risk-based MFA work is the unique combination: a composite score, set by your organization and derived from all signals evaluated together.

Any single HIGH-risk signal elevates the composite. As an example “impossible travel” alone could be enough to trigger a challenge. But, multiple moderate signals can also combine into a composite that warrants a challenge even when no individual signal is alarming on its own. A login from an unfamiliar device, with a stale password, after a six-month absence: each signal is ambiguous in isolation; together, they tell a different story.

Don’t Overlook

Not all providers check for all ten factors. Make sure this is a question when conducting vendor evaluations.

Here’s what these signals might look like in practice.

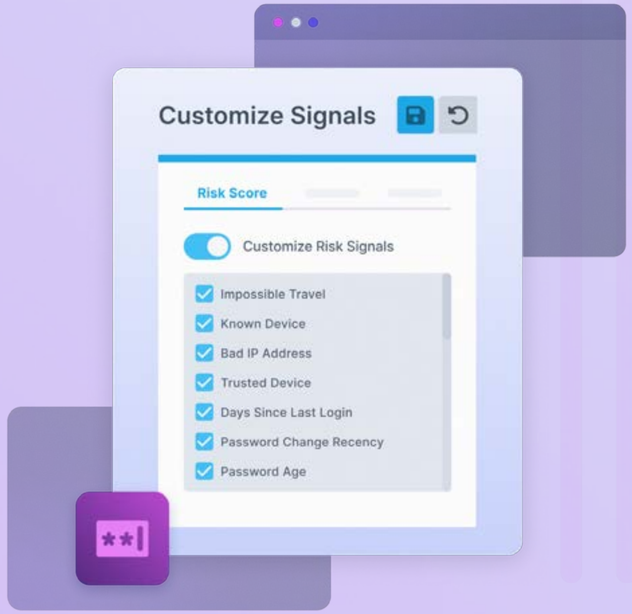
Example Login	Risk Signal & What Happens
8:47 AM Your usual phone, home Wi-Fi, morning routine. Recognized device, clean network, normal hours, no recent account changes.	**COMPOSITE SCORE: LOW** You’re in, no extra step required. This is the goal: invisible security when nothing looks wrong.
1:15 PM Same account, same phone, but you’re on a public coffee shop network flagged for suspicious activity. Everything else checks out.	**COMPOSITE SCORE: MEDIUM** Depending on how the app is configured, you might get a quick verification prompt or sail straight through. One yellow flag doesn’t always mean a roadblock.
11:45 PM Login attempt on your account from an IP in another country, on a device you’ve never used. If you logged in from home earlier today, impossible travel applies. Unknown device. Unusual hour.	**COMPOSITE SCORE: HIGH** You get a verification challenge automatically. If it’s really you traveling, a quick tap confirms it. If it’s not you, the attacker is stopped cold.

Building a Risk-Based MFA Policy

An MFA policy is only as effective as the thinking behind it.

Deciding when to challenge users, how aggressively to respond to risk signals, and what exceptions to allow requires deliberate choices that vary by environment, user base, and threat model.

Those choices get more consequential for companies that market their products or services directly to consumers.



You're not managing a few thousand employees on a corporate network. You're protecting millions of accounts tied to real identities: payment methods, health records, financial data, personal communications, or simply years of trust built between a user and your product. Attackers know this. Credential stuffing campaigns target consumer apps specifically because the volume of reused passwords is high and the payoff per compromised account is real. And unlike enterprise users, your customers can walk. An employee who hits a friction wall has nowhere else to go. A shopper or subscriber who finds account access annoying will find a competitor who makes it easier.

The tension between security rigorous enough to stop attacks and UX smooth enough to retain customers is exactly what a well-designed MFA policy has to resolve.

Set Risk Thresholds Deliberately

A consumer product rarely lives on a single surface. Web app, mobile app, billing portal. They all run on the same identity platform, but they don't carry the same risk. Even within a single app, actions like browsing a product page carry a significantly different risk than changing an account email or initiating a transfer.

A blanket challenge policy ignores that distinction. Apply it too aggressively and you frustrate users on low-risk flows. Apply it too loosely and you leave your highest-stakes moments exposed.

The better approach is a sensible default with targeted exceptions. Set your baseline to challenge on HIGH-risk logins across all apps, then tighten specific flows where the stakes are higher, such as your payments portal, which might require a challenge at a lower risk threshold than your content app. One policy foundation, adjusted where it matters.

The Policy Decision Tree

A workable MFA policy has three tiers.

1. LOW-RISK SESSIONS:
Pass through without a challenge. Define what low risk means for your app (recognized device, clean IP, a recently active account) and hold the line. Friction here has a real cost: abandoned signups, abandoned carts, and a barrage of support tickets all trace back to frequent and unnecessary challenges.

Deliberate Decisions

The policy only works if the thresholds are deliberate. ‘We challenge on HIGH” is a decision. ‘We challenge on HIGH and MEDIUM for admin consoles but HIGH only for standard applications’ is a better one.

2. MEDIUM-RISK SESSIONS:
Challenge with whatever method the user has enrolled. This tier catches the “something is different but not definitively wrong” cases. The key decision here is whether MEDIUM risk triggers a challenge at all or whether you only challenge on HIGH. Make that decision explicitly and document it. This is the main lever between security and friction for your customer base.

3. HIGH-RISK SESSIONS:
Always challenge, no exceptions. Think of an unrecognized device coming from a flagged IP, or a long dormant account suddenly logging in and immediately changing a password. Those types of login attempts should never slide through. For the most sensitive of these, you may also want to require a step-up even after a successful login (more on that below).

Make Risk Decisions Explainable

There’s an important architectural choice buried in how risk-based MFA systems work: whether the risk engine uses machine learning or deterministic rules. An ML-based approach may catch patterns a rules-based system misses, but it’s a black box. That can come with a real cost when it comes to consumer products.

Your risk engine will sometimes challenge or block a legitimate paying customer. When that happens, you need to know exactly why so you can tune the system. With a black-box model, that answer isn’t available. You can’t explain to a frustrated customer why they were stopped, you can’t tell your support team what triggered the challenge, and in regulated industries, you can’t satisfy an auditor asking why a specific user was flagged at a specific moment.

A rules-based engine, where every challenge decision traces to specific named signals, each mapped to a specific risk level, gives you something you can explain, document, and defend.

Add Step-Up Authentication Where the Stakes Are Higher

One of the most underused capabilities in designing an MFA policy is enabling a step-up authentication that triggers a fresh challenge when a user attempts a specific high-risk action within an authenticated session. This takes the protection beyond the login to the action the consumer is attempting.

Consider these consumer use cases: re-verify before changing an account email or password, before adding or changing a payout or withdrawal method, before a high-value purchase, or before exporting personal data.

A session that was perfectly legitimate at login can become hostile by the time someone tries to move money or change a recovery email, which is exactly what an attacker who's hijacked a live session does. Step-up authentication puts a checkpoint at the moments that matter most.

Design Recovery Before Users Get Locked Out

When an employee loses their phone, they can walk over to IT and prove who they are in person. Customers can't. They're stuck with whatever self-service recovery you built, or are relegated to the dreaded support queue. That's a problem.

If your recovery process is "contact support and wait for an agent to turn off MFA for you," two outcomes are likely and both of them are bad. Customers get frustrated waiting and may not return. Worse, the attacker contacts support pretending to be locked-out and talks their way past the very protection you set up.

The easiest solution for this is to make sure that your policy design accounts for it from the start. First, encourage each user to enroll in more than one method (email, SMS, and TOTP), so losing one doesn't cause a lockout. Second, issue recovery codes at enrollment. Lastly, build a verified, self-service recovery flow for the cases where someone loses everything (it will happen).

Use Remembered Devices Without Ignoring Risk

"Remember this device" is one of the biggest friction reducers you have for returning customers, so it's worth getting right. A trust token stored on the device signals prior verification and lets you challenge known hardware less often. Give that trust an explicit expiration, since a device trusted ninety days ago doesn't deserve the same benefit of the doubt as one trusted last week.

What Takes Precedence

Device trust should not override a risk-based challenge in high-risk scenarios. If a session scores HIGH, the fact that the device was previously trusted shouldn't suppress the challenge. The risk score and the device trust record are separate inputs; risk should take precedence when they conflict.

Making Risk-Based MFA Work With Your Security Stack

MFA doesn't produce useful signals if the data it generates stays inside your authentication platform.

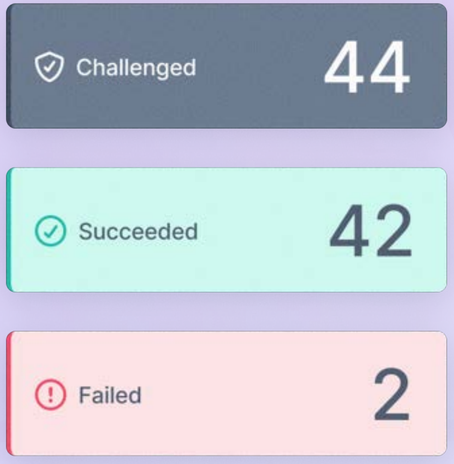
Every challenge, every failure, every success is an event your security operations team should be able to see, correlate, and act on. For a consumer app, those signals are how you catch account takeover early and warn a customer before real damage is done. The organizations that get the most security value from their MFA deployment are the ones who've connected it to the rest of their stack.

Most auth platforms can send these events to other systems, usually through webhooks. You point them at an endpoint, and your monitoring, fraud, or alerting tools pick them up. Some events are worth acting on automatically. Some are worth surfacing to your team. And some are worth telling the customer about directly.

The events fall into two groups: ones that track normal MFA activity, and ones that hint at a compromised account.

1. Normal MFA activity

Most of what MFA produces is routine, but capturing that routine is what makes the rest possible. Log every challenge along with the risk score behind it, and keep track of whether each one passes or fails. The value is in the pattern more than any single event. A lone failed attempt is almost always a mistyped code, while a rapid run of failures against one account is a machine trying its luck. That's your cue to lock things down. Once you're recording all of it, you have a baseline for what normal looks like, which is exactly what you need in order to spot when something isn't.



Your security operations team should be able to see, correlate, and act on every login attempt.

2. Signs of a compromised account

Some events are quieter but more telling. They're the fingerprints of an account takeover in progress.

A high-risk login score is the earliest signal, and worth capturing even when no challenge fires. A trusted device can wave a suspicious session straight through, so logging the risk score matters even when authentication succeeds.

A login from an unfamiliar device rarely means trouble on its own, but paired with any other elevated signal it's worth a quick "was this you?" to the customer.

The most overlooked signal is also the most dangerous. An attacker who gets into an account will often remove the real owner's second factor first, locking them out and clearing the path for everything that follows. Treat MFA method removal as a genuine alarm. Flag it immediately and reach the account owner through a separate channel to confirm they made the change.

Event	What It Means	Suggested Response
Challenge triggered	Risk engine flagged at login	Log it; record the risk score
Challenge failed	A second-factor attempt failed	Watch for repeated failures, lock after a threshold.
Challenge succeeded	Login completed with MFA	Log for audit trail and records; use as baseline
High risk login	Session scored high on the risk signal	Review it; consider alerting customer
Login from a new device	Unrecognized device	Notify user out-of-band; correlate with other signals
Second factor removed	MFA increasingly required for production access	Immediate alert; verify with account owner via separate channel

Rate limiting as a defensive layer

One operational control pairs well with all of this: rate limiting on login endpoints.

Credential-stuffing tools are designed to send authentication requests at scale, thousands of attempts per minute. A rate limit that caps failed attempts per IP or per account over a rolling window stops automated attacks before they become a flood. It protects customers and cuts the noise the team has to sit through. It's a simple control with an outsized effect on automated attacks.

Choose Where Your Risk-Based MFA Runs

For most software, “Cloud or self-hosted” is an IT preference.

For consumer identity, it’s a product and compliance decision, because it determines where your customers’ personal data physically sits, how fast their logins resolve, and what you pay as you grow into the millions.

At consumer scale, deployment decisions show up in three places: compliance, login performance, and cost.

DATA RESIDENCY IS INCREASINGLY THE LAW

GDPR in the EU, and a growing list of data-localization rules in markets like India and Brazil, expect that customer PII to stay in-region. If your identity platform doesn’t support the regions you need, meeting data residency requirements becomes an architectural problem you have to solve elsewhere.

LOGIN IS HIGH-VOLUME AND LATENCY SENSITIVE

Authentication sits in front of every session. Running it close to your users is a performance concern at consumer scale in a way that it never is for a few thousand employees.

PER-USER PRICING PUNISHES CONSUMER VOLUME

Many cloud-based CIAM platforms bill per monthly active user, and often gate their risk-based MFA behind a higher tier or make it an add on. At consumer scale, that model can dominate your bill and quietly push your teams toward running identity themselves.

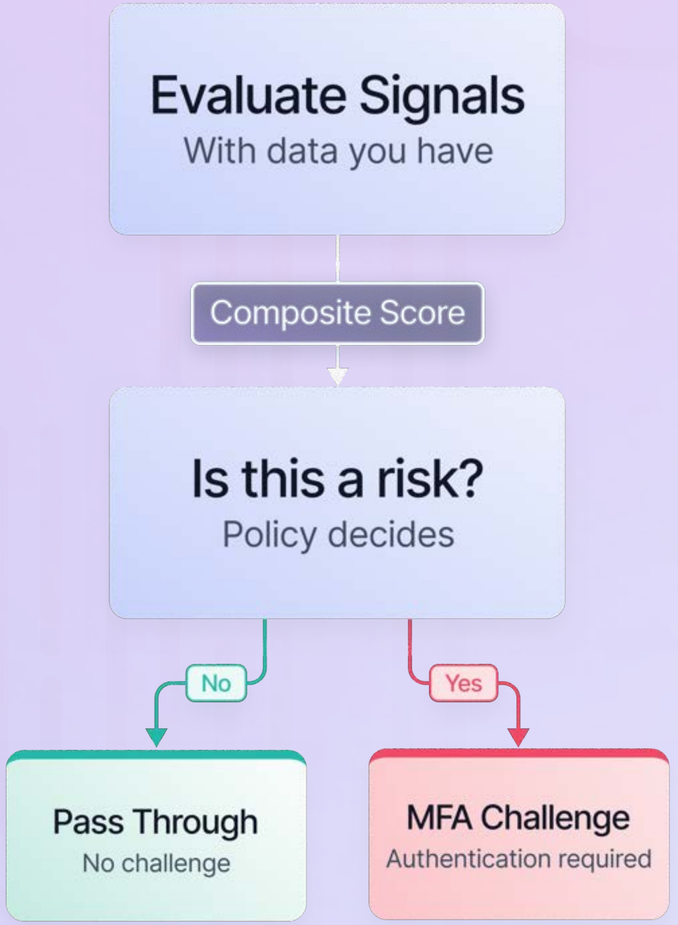
Don’t Let Deployment Limit Your Risk Engine

The catch most vendors don’t advertise is that many risk-based MFA engines only run in the vendor’s cloud. So, if residency law or scale pushes you to self-host, you’re forced to choose between running identity on your own infrastructure and not having access to risk-based MFA at all.

None of that is inevitable. It’s a choice vendors make when they build the risk engine to run only in their own cloud. Most of the signals that matter come from data you already hold: whether the device is familiar, or how recently the user logged in, how old their password is. Only a couple, IP reputation and known-bad user agents, depend on a live threat feed and need connectivity to stay recent. Build it right and those drop to a MEDIUM risk score when the feed can’t be reached, rather than locking a real customer out.

That changes what’s possible. If the engine scores risk from local data, there’s no technical reason the whole flow can’t run inside your own environment: your cloud account, your data center, or a network with no outbound access at all. Customer login data never leaves your boundary, residency requirements take care of themselves, and there’s no per-user surcharge for switching risk scoring on.

What you take on in return is ownership. Self-hosting means you run the upgrades, scale the service to your traffic, and, if you want the threat-intelligence signals at full strength, keep an outbound path open to the feed. For a lot of consumer teams that’s a fair trade, but it’s a decision you need to consciously make, not one that a vendor forces you into.



Getting Customers to Actually Adopt MFA

In a consumer product, MFA rarely fails because the technology breaks. It fails because people abandon it. Push a second factor on customers the wrong way and they quit the signup and leave. The teams that get this right roll it out gradually, make enrollment effortless, and lean on risk so most logins don't feel a thing. A phased approach reduces risks.

A Phased Approach to MFA Adoption

PHASE 1:
KNOW WHERE YOU STAND

Before you change anything, know what you have. How many accounts already have a second factor and which methods do most customers choose? Where does the risk concentrate in the accounts that hold payment details, balances, or personal data? You can't improve an adoption rate you've never measured.

PHASE 2:
ROLL OUT TO A SUBSET OF USERS FIRST

Instead of turning MFA on for everyone at the same time, release it to a small percentage of users or a lower-stakes segment. Watch what actually happens. How many finish enrollment, how many drop off, and how many contact support? Fix the friction there before it reaches millions of users.

PHASE 3:
PROTECT THE HIGHEST-VALUE ACCOUNTS FIRST

The accounts worth attacking are the ones with stored payment methods, balances, or control over other users. Offer or require a second factor to those first. Those customers have the most to lose, so they're likely willing to accept a little friction to stay safe.

PHASE 4:
MAKE ENROLLMENT EFFORTLESS AND WORTH IT

This is where consumer rollouts are won or lost. The goal is to make enrolling feel easy and obviously worth it: ask at a sensible moment (not mid-checkout) and explain in plain terms how MFA keeps their account safe. And, of course, make sure that setup is easy. A clear reason and a fast setup will always beat a mandate.

PHASE 5:
SWITCH ON RISK-BASED CHALLENGES, THEN WATCH THE NUMBERS

Rather than challenge everyone at every login, let the risk engine decide. Routine logins stay invisible and a step-up kicks in only for a suspicious session or sensitive action. Then track what matters like adoption, how often they fail (a sign of a broken method or confusing screen), and the business signals of abandonment, support contacts, and churn.

Moving from another login system

If you're migrating from an existing platform, don't force millions of customers to set up their second factor again. For a consumer base, that will create churn. Some providers, like FusionAuth, make it possible to import existing TOTP secrets and phone numbers through an API, so established methods carry over and the change stays invisible to the customer. Confirm your new platform actually carries the enrolled MFA data before you commit, because some import the users but leave the second factors behind, which forces everyone to re-enroll anyway.

Build Recovery Into Your MFA Policy

At enrollment, every customer should receive recovery codes: backup, single-use codes that work when the primary MFA method is unavailable. Tell users to store them somewhere safe, in writing, offline. Then, decide in advance what a customer does once they've burned every code and lost every method.

For a consumer product, that fallback has to be self-service and has to verify identity, because a human who can simply switch MFA off is the exact hole attackers aim for. Build that path before you make MFA required, not after the first customer is locked out for good.

Handling Customers Objections

Objection	Response in product and in policy
"Ugh, another step just to log in?"	With risk-based MFA, most logins aren't challenged at all. You'll usually only be asked when something looks off, like a new device or an unfamiliar location.
"What happens if I lose my phone?"	The one that matters most. Let people enroll in more than one method, hand out recovery codes at setup, and offer a self-service way to get back in that doesn't rely on reaching a human.
"I don't want to use my personal phone for work."	Don't force SMS. Offer an authenticator app or email as alternatives, and be clear the number is only used to protect the account.
"Why do I suddenly need this?"	Frame it as protecting their account and the value that sits inside it. Account takeover is rising, authentication is how you keep them out.
"What if I'm traveling internationally or somewhere with no signal?"	An authenticator app generates codes offline, and recovery codes work with no connection at all.
"What happens when I get a new phone?"	Most authenticator apps support account transfer or backup, but the process needs to be documented before users hit it.

Pair Passkeys With Risk-Based MFA

The hardest thing to phish is a credential that's cryptographically bound to the specific website or device it was created for. That's the core of how passkeys work. When someone creates a passkey for your app, the keys are bound to your domain, so a fake lookalike site can't use them, copy them, or replay them. The device confirms the person with a fingerprint, face, or PIN, proves it holds the key, and the login completes. Nothing secret crosses the network, so there's nothing for an attacker in the middle to steal.

For consumer apps, what changed is reach. Passkeys used to be impractical for a mass audience. Now Apple, Google, and Microsoft have built them into their phones, browsers, and operating systems, so most of your customers already have everything they need, with no extra app to install. That makes passkeys a realistic default for consumer login, not just a security-team luxury.

Passkeys and risk-based MFA aren't competing choices. They work together. A passkey login already carries strong signals: a known device, a phishing-resistant method, and the user's presence confirmed in the moment. A risk engine can read those signals and let the session through with little or no added friction, while still watching for anything that looks wrong. Strong login methods lower the baseline risk. They don't remove the value of paying attention to context.

The practical move for most consumer apps is to start offering passkeys as a login option, even if passwords stick around a while longer, and to understand how your risk engine treats a passkey login as an input. That's the foundation for a modern, low-friction login experience.

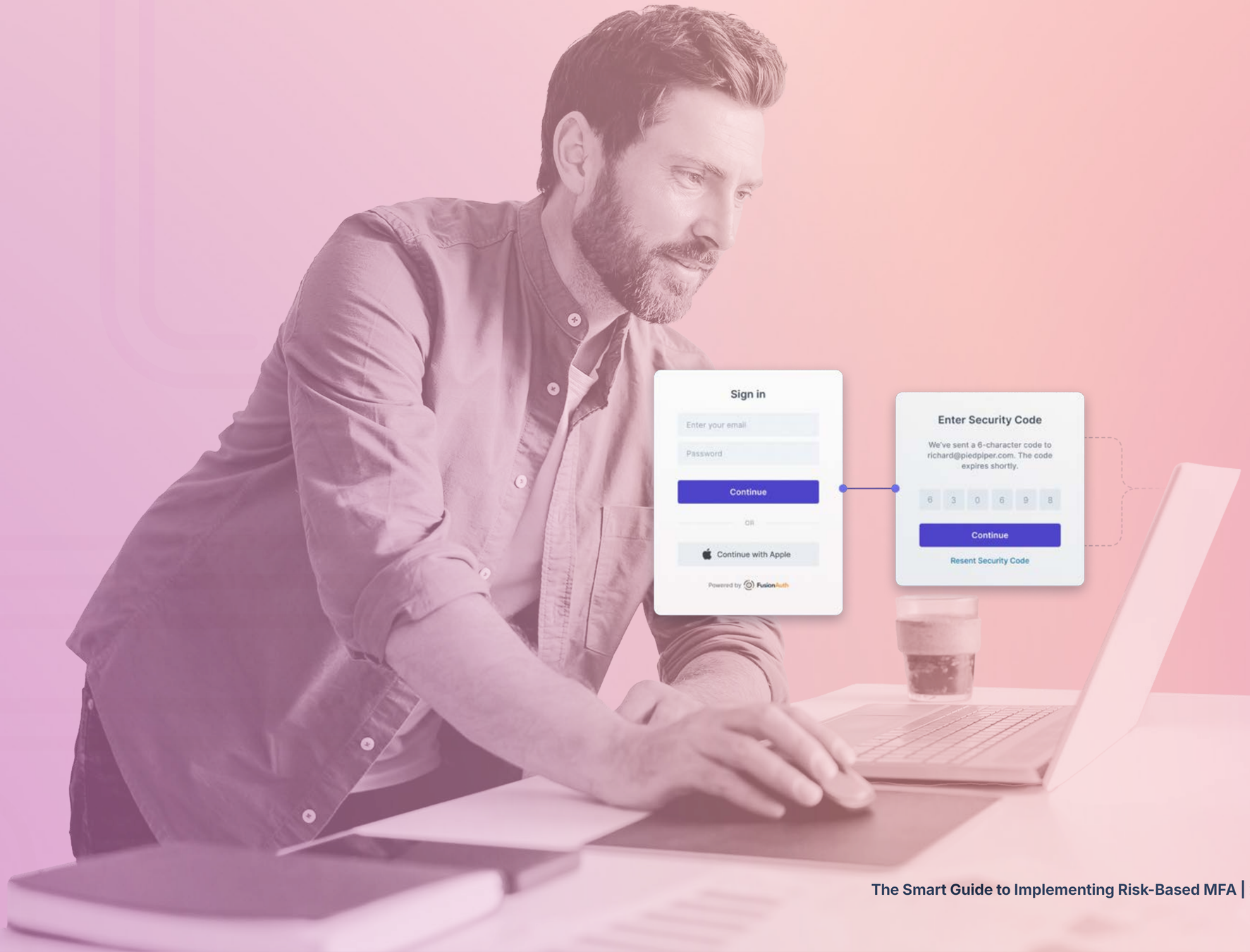
For security leaders, the practical implication is this: passkeys are the category of MFA that closes the AiTM phishing gap. Users can still be socially engineered into doing things they shouldn't, but the credential they're using with passkeys can't be stolen and replayed on a different site by a proxy in the middle.

The Passkey Adoption Picture Today

The adoption friction that once made phishing-resistant passkeys impractical at scale has dropped substantially. Apple, Google, and Microsoft have all built passkey support into their platforms at the OS and browser level. A user on a modern iPhone, Android device, Mac, or Windows machine already has the infrastructure to register and use passkeys, no additional app or hardware required. Hardware roaming keys like YubiKey remain the right choice for high-security environments where you want a physical token that isn't tied to a specific device, but for broad enterprise deployments, the platform-native passkey experience has become genuinely practical.

SECTION 04

Closing



Risk-Based MFA Is a Strategy, Not a Feature

Done well, MFA does more than verify identity. For a consumer product, it's a live signal layer that catches account takeover before it costs the customer (and your business). It's a record you can inspect and explain when a real user gets blocked. It's less friction for the vast majority who just want to log in, with resistance applied exactly where attackers probe. It also holds up across millions of logins, instead of treating every one as equally suspicious.

The organizations that get this right have a measurable advantage: fewer account compromises, faster incident detection, cleaner compliance audits, and a security posture that adapts as the threat landscape evolves. Organizations that treat MFA as a checkbox are missing out and leaving customers exposed.

Here are four things you can do right now to evaluate your security posture:

1. OFFER CUSTOMERS REAL FACTOR CHOICES
Look at the factors you offer. Do customers have real choices across an authenticator app, SMS, and email, and does SMS have a stronger backup for the accounts that hold real value? Leaning on a single factor for everyone is where friction and risk both pile up.

2. CHALLENGE HIGH-RISK ACCOUNT ACTIONS
Identify the actions that deserve a fresh challenge mid-session. Changing the account email or password, adding a payout method, moving money, exporting personal data. These are the moments takeover actually shows up.

3. CHOOSE THE RIGHT DEPLOYMENT MODEL
Evaluate whether your authentication platform supports the deployment model your environment requires. In regulated industries with data residency obligations or air-gapped network segments, the ability to run MFA on your own infrastructure is a requirement that should drive your platform evaluation.

4. START CAPTURING MFA EVENTS
Start capturing MFA events. The high-risk logins, the repeated failures, and especially the second-factor removals that precede a takeover all exist if your platform makes them available. If that data isn't reaching your fraud and monitoring tools, or triggering a heads-up to the customer, you're sitting on one of your best early warnings.

Putting These Principles Into Practice with FusionAuth

FusionAuth is built around the principles discussed throughout this guide: flexible deployment, risk-aware challenges, self-service recovery, and events that connect to the rest of your security stack.

Attackers adapt, privacy requirements change, and your customer base grows. The right foundation gives your MFA strategy room to evolve without forcing you to re-architect it every time the bar moves.

MORE CONTEXT LEADS TO BETTER DECISIONS

FusionAuth evaluates every login against ten independent risk signals, including device familiarity, IP reputation, password age, login recency, impossible travel, suspicious browser fingerprints, and dormant account reactivation. Most competitors evaluate only a few. More signals mean better context and risk accuracy, fewer unnecessary MFA prompts for legitimate users, and greater confidence that the logins you challenge truly deserve it.

AUTHENTICATION DECISIONS SHOULD BE EXPLAINABLE

FusionAuth uses a deterministic risk engine rather than an opaque machine learning model. Every challenge traces back to specific named signals and configurable risk thresholds, so when a legitimate customer is challenged unexpectedly, you can see exactly why. That makes it easier to tune policies, investigate false positives, support customers, and satisfy auditors without relying on a black-box model that can't explain its own decisions.

RISK-BASED MFA SHOULDN'T BE RESERVED FOR PREMIUM TIERS

FusionAuth includes risk-based MFA starting at the entry tier, with no separate attack protection add-on and no per-monthly-active-user surcharge to enable it. At consumer scale, security pricing matters. Identity platforms shouldn't force you to choose between stronger authentication and predictable costs as your customer base grows.

DEPLOYMENT SHOULD BE YOUR DECISION, NOT YOUR VENDOR'S

Whether you deploy FusionAuth in our cloud, your own cloud account, your own data center, or a fully air-gapped environment, the same risk engine is available. That means compliance requirements, data residency, latency, and operational preferences can drive your deployment strategy instead of limitations in your identity platform.

FusionAuth | Ebook

MFA isn’t a problem you solve once. Attackers adapt, privacy rules change, and your customer base grows. Build the right foundation now, with flexible deployment, risk-aware challenges, self-service recovery, and events that connect to your systems, so you can keep pace as the bar moves instead of re-architecting each time.

Interested in learning more about how risk-based MFA can help reduce friction while improving security outcomes? Contact FusionAuth at https://fusionauth.io/contact