



DMARC, Explained

How to stop other people sending mail
that looks like it came from you

A plain-English guide for people who run businesses, not mail servers. The third in a series — the first two cover SPF and DKIM. Updated for the 2026 revision of the standard, which removed a tag most guidance still recommends.

01

The problem DMARC solves

SPF and DKIM both check names your reader never sees.

SPF checks the computer that handed the message over. DKIM checks that the message arrived as it was sent. Both work. Both can pass on a message that is pretending to be from you.

The reason is that an email carries more than one sender name, and the one your reader actually sees is not the one SPF and DKIM were checking.

WHAT YOUR READER SEES

Your Bank

security@yourbank.com

Please confirm your account details

WHAT SPF AND DKIM WERE CHECKING

SPF looked at	bounce@mailers-7.example.net	pass
DKIM looked at	example.net	pass

Both passed. Neither had looked at the name on the From line. That is the gap DMARC closes.

Figure 1 — Two passes, and nobody checked the name on the screen.

A forger can register a domain, publish a perfectly valid SPF record for it, sign their mail with a perfectly valid DKIM key, and put *your* name on the From line. Every check passes. The message is still a forgery.

What DMARC actually is

A rule you publish saying: a message claiming to be from my domain must have passed SPF or DKIM **using my domain**, not somebody else's. And here is what to do with it if it did not.

Alignment: the whole idea

One comparison. Everything else in DMARC is detail hanging off it.

An email carries three domain names that matter here. Your reader sees exactly one of them.

DMARC asks one question: does a name that passed match the name on the From line?

The From line	yourbank.com	the only name your reader sees
The SPF name	mailer-7.example.net	hidden in the envelope
The DKIM name	example.net	hidden in the signature

When a passing name matches the From line, that is alignment, and DMARC passes. Neither of the hidden names above matches yourbank.com, so this message fails DMARC — even though SPF and DKIM both passed.

Figure 2 — Alignment is a comparison, not a check.

That comparison is called **alignment**, and it is the thing DMARC adds. SPF and DKIM produce a pass or a fail; DMARC asks whether the name that passed is the name your reader is looking at.

Why a message can pass SPF and DKIM and still fail DMARC

Because passing is not the same as passing as *you*. This is the single most common surprise when people first read their DMARC reports, and it is not a fault — it is the protocol working. A newsletter tool that signs with its own domain will show exactly this pattern.

The two ways to pass

A message needs one of them. Not both.

Either route is enough on its own. A message needs one, not both.

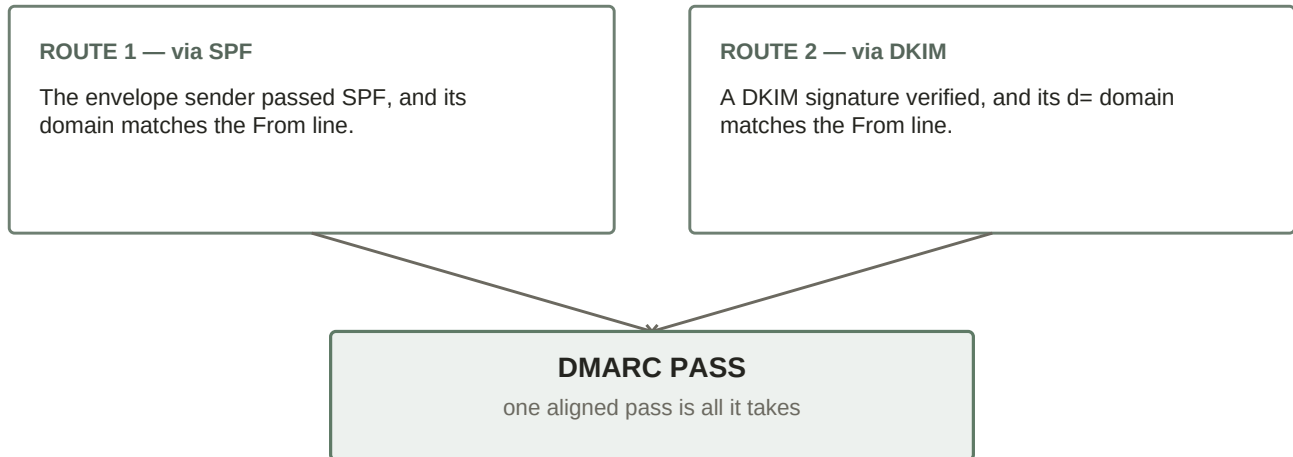


Figure 3 — Either lane gets you there.

This matters more than it sounds. Forwarding usually breaks SPF, because the forwarding server is not on your list. If the DKIM signature survives — and it usually does — the message still passes DMARC by the other route.

That redundancy is the practical argument for setting up both SPF and DKIM rather than picking one. Each covers the other's bad days.

One detail worth knowing

A message can carry several DKIM signatures. DMARC passes if *any one* of them both verifies and aligns with your From line. The others failing does not matter.

Relaxed and strict matching

A choice you will probably never need to make.

When DMARC compares two names, it can be generous or literal about what counts as a match. Generous is the default.

Relaxed is the default, and is what nearly everyone should use.

The name that passed	The From line	Relaxed	Strict
mail.yourbank.com same organisational domain	yourbank.com	pass	fail
yourbank.com identical	yourbank.com	pass	pass
yourbank.example.net different domain entirely	yourbank.com	fail	fail

Strict buys very little and breaks mail sent from subdomains. Unless you have a specific reason, leave adkim and aspf alone.

Figure 4 — Relaxed accepts a subdomain; strict demands the exact name.

Relaxed matching treats mail.yourbank.com and yourbank.com as the same organisation, which is almost always what you want. Strict matching requires the two names to be identical, character for character.

Strict is included here only so that you recognise it if you meet it. Turning it on tends to break mail sent from subdomains by systems you had forgotten about, in exchange for protection against an attacker who has already compromised a subdomain of your own domain — at which point you have a larger problem.

What you are asking receivers to do

Three settings, and only one of them stops anything.

Three settings. Only the last one actually stops anything.

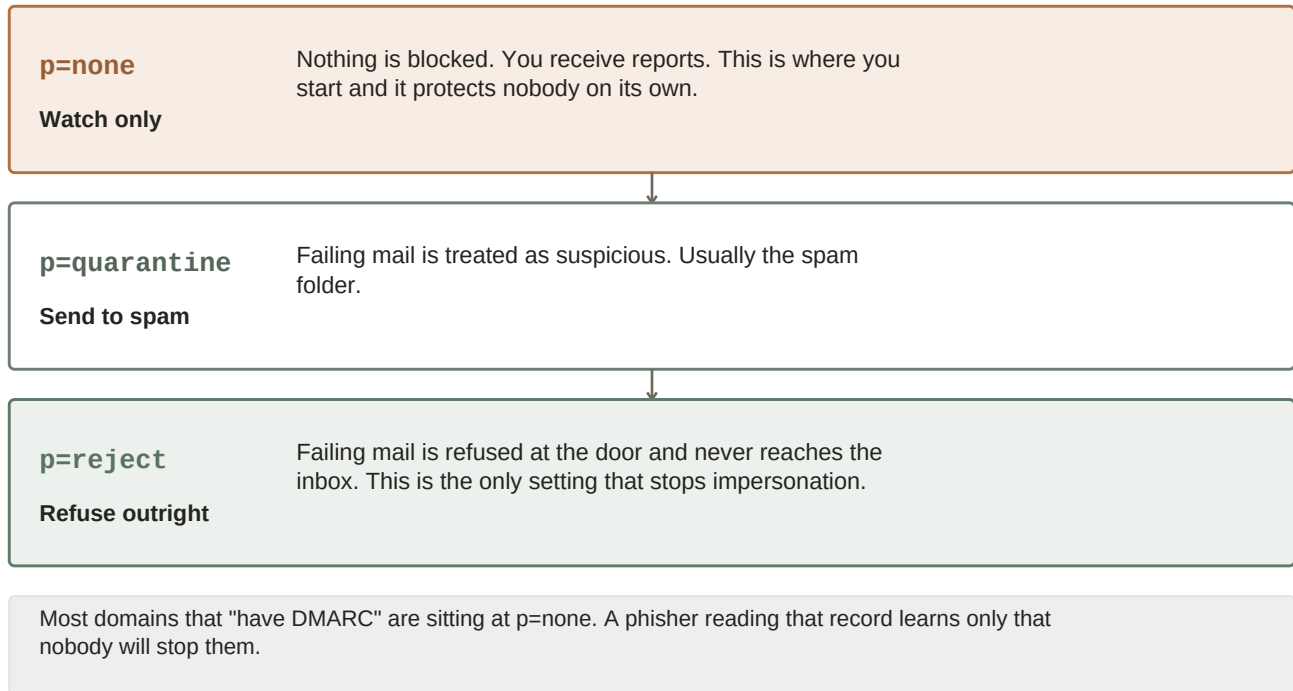


Figure 5 — The policy ladder.

It is worth being precise about what this is. You are publishing a *request*. Receivers are not obliged to honour it, and you have no way to make them. In practice the large mailbox providers do honour it, which is what makes DMARC useful — but the record is a preference, not a control you hold.

p=none protects nobody

It is a monitoring setting. It exists so you can see what is happening before you change anything, and it is the correct place to start — but a domain parked at p=none for two years is a domain anyone can still impersonate. Publishing it is the beginning of the work, not the end of it.

Reading a DMARC record

One line of text, published once, in your DNS.

```
v=DMARC1; p=reject; rua=mailto:dmarc@yourbank.com;
sp=reject; np=reject
```

v=DMARC1

The version. Always this, and it must come first.

p=reject

What you are asking receivers to do with mail that fails. The setting that does the work.

rua=mailto:...

Where the daily summary reports are sent. Without this you are flying blind.

sp=reject

A different policy for existing subdomains. Optional.

np=reject

A policy for subdomains that do not exist at all. Cheap protection, added in the 2026 revision.

adkim=r aspf=r

Relaxed or strict matching. These are the defaults; you can usually omit them.

Published as a TXT record at _dmarc.yourbank.com. Nothing in it is secret.

Figure 6 — The record, pulled apart.

If you only ever read two, read p= and rua=. The first is what you are asking for. The second is how you find out whether asking for it is safe yet.

Also seen	What it means
t=y	Test mode. Receivers apply one step below your stated policy — reject behaves as quarantine, quarantine as none. Added in 2026.
fo=	Which kinds of failure should produce a detailed report. Ignored unless you also publish ruf=.
ruf=	Where to send per-message failure reports. These can contain message content, so think about privacy first.
psd=	Marks a record as belonging to a public suffix operator. Not something an ordinary domain sets.

Subdomains, including ones that do not exist

The gap most records leave open.

Your policy covers your domain and, by default, its subdomains. But there are two separate cases, and you can set them independently.

- **Subdomains that exist.** Real ones you send from — `mail.yourbank.com`, `news.yourbank.com`. The `sp=` tag sets their policy. Note that it applies to the subdomains, not to the domain itself.
- **Subdomains that do not exist.** A forger does not need a real subdomain — `accounts.yourbank.com` works just as well as a From address whether or not you ever created it. The `np=` tag covers exactly this case.

np=reject is close to free

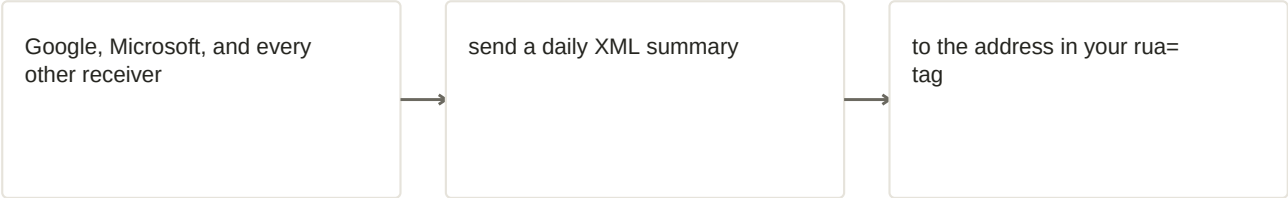
By definition it applies only to names you never set up, so it cannot break any mail you are actually sending. If you publish nothing else from this guide, publishing `np=reject` costs you nothing and closes a real avenue of attack.

Receivers find the right record by starting at the exact name on the From line and walking up towards the registered domain until they find one. The 2026 revision made this walk explicit and capped it at eight lookups, replacing an older method that relied on every receiver keeping the same list of domain suffixes up to date.

The reports, and why they come first

The part people skip, and the reason rollouts fail.

Reports are how you find the senders you forgot about.



A typical week of that mailbox, once decoded:

your mail provider	41,020 messages	aligned pass
your newsletter tool	6,400 messages	aligned pass
an invoicing system nobody mentioned	380 messages	failing
a host in another country	2,900 messages	failing

The third line is why you do not jump straight to p=reject. The fourth is why you do not stay at p=none.

Figure 7 — A week of aggregate reports, decoded.

Aggregate reports are daily XML summaries from receivers, listing who sent mail as you and how it fared. They contain no message content — just counts, sending addresses and results. Raw, they are close to unreadable; any DMARC tool, including ReaperCloud's, will turn them into the kind of table above.

If you publish no rua= address, receivers are required not to send you these reports at all. You then have no way of knowing what moving to p=reject would break.

What you are looking for

Not the forgeries — those are the easy part. You are looking for your own systems that fail: the invoicing tool, the recruitment platform, the office printer. Every one of those is mail that disappears the day you move to reject, unless you fix it first.

Rolling it out without blocking your own mail

The order is the whole technique.

The order matters. Every step exists to stop step 4 blocking your own mail.

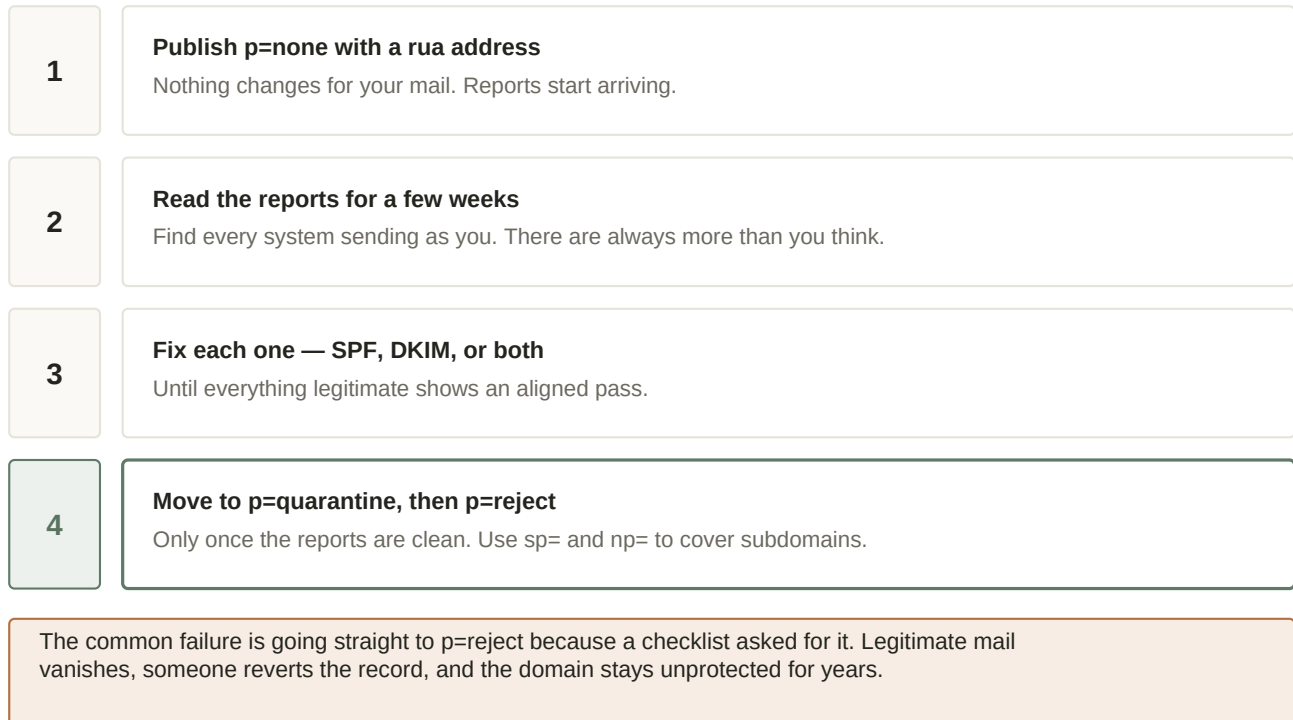


Figure 8 — Four steps, in this order.

There is no prize for reaching p=reject quickly, and there is a real penalty for reaching it before the reports are clean: invoices that never arrive, password resets that vanish, and a colleague who reverts the record and never touches it again.

A few weeks at each stage is normal. Large organisations take months, because the hard part is not the DNS record — it is finding every department that quietly signed up for a mail service.

What changed in 2026

Why this guide disagrees with a lot of advice online.

DMARC was published as an informational document in 2015 and became a formal standard in May 2026. The revision made two changes an ordinary domain owner will actually notice.

Change	What it means for you
pct= is gone	It used to apply your policy to a percentage of mail. It was widely misunderstood, inconsistently implemented, and has been retired. If your record still contains it, receivers now ignore it — which may mean your policy is being applied far more broadly than you intended.
t=y replaces it	For a gentler rollout, t=y asks receivers to apply one step below your stated policy. Unlike pct=, it applies to all mail rather than a random slice, so what you observe is what you get.

If you take one action from this page

Look at your current record. If it contains pct=, decide deliberately what you want to happen now that it is being ignored — either remove it and accept full enforcement, or step the policy down while you finish the work.

A short checklist

What to publish, and what to look for in the reports.

Getting it right

- **Start at p=none with a rua address.** Never the other way round.
- **Set up SPF and DKIM first.** DMARC has nothing to work with otherwise, and a domain with neither will fail every message the moment you enforce.
- **Add np=reject early.** It cannot break mail you send, because it only covers names that do not exist.
- **Read the reports before every change.** The reports decide when you move, not the calendar.
- **Leave adkim and aspf alone** unless you have a specific reason.

Signs something is wrong

- Your record still contains pct=. It is now ignored — see Section 10.
- You publish a policy but no rua=. You have turned off your only source of evidence.
- You have been at p=none for over a year. The domain is no better protected than one with no DMARC at all.
- Reports show a sender you cannot identify passing alignment. Someone in your organisation set something up — find out who before you assume it is an attack.
- Your policy is reject but np= is unset and sp= is looser. Check that is deliberate.

Check your own domain

ReaperCloud's DMARC tool reads your record, flags retired tags, and shows what your policy actually resolves to for subdomains. `reapercloud.com/dmarc.php`, or `reapercloud.com/report.php` for SPF, DKIM and DMARC together. The reports page turns aggregate XML into something readable.

This document describes DMARC as specified in RFC 9989, which replaced RFC 7489 in May 2026, with aggregate and failure reporting defined in RFC 9990 and RFC 9991. It is an explanation, not a specification: where they differ, the RFCs are correct and this is simplified.