



MTA-STS, Explained

How to stop your mail being read
on its way to the people you send it to

A plain-English guide for people who run businesses, not mail servers.
The fourth in a series — the first three cover SPF, DKIM and DMARC.
This one is different: it is not about who sent a message, but about the
connection that carries it.

01

A different kind of problem

Nothing in this guide is about forgery.

SPF, DKIM and DMARC all answer versions of the same question: is this message really from who it says it is? This one does not. It assumes the message is genuine and asks whether anyone could read it on the way.

Mail servers encrypt their connections to each other using STARTTLS. The weakness is in how that gets agreed: the receiving server *offers* encryption, and the sending server accepts. An offer made in the open can be deleted by anyone sitting in the middle.

Nothing is forged. The message is genuine. It is simply read on the way.



Your server sees a mail server that does not support encryption

So it does the helpful thing and sends anyway, in plain text. There is no error, no warning, and nothing in the message to suggest anything went wrong. SPF, DKIM and DMARC all still pass — they were never looking at the connection.

MTA-STS is how you tell senders in advance: my servers do support encryption, so refuse to believe anyone who says otherwise.

Figure 1 — The offer is removed, so the sender never knows it was available.

What MTA-STS actually is

A statement you publish saying: my mail servers are these, they all support encryption, and if anyone tells you otherwise they are lying — do not deliver. Sending servers read it in advance and remember it.

What it does not do

Worth settling early, because the names all look alike.

MTA-STS does nothing whatsoever about phishing, spoofing or spam. A forger who sends mail pretending to be you is unaffected by it. If that is the problem you are trying to solve, the DMARC guide is the one you want, and it should come first.

If you want to stop...	Use
People sending mail that looks like it came from you	SPF, DKIM and DMARC
Your mail being read or altered between servers	MTA-STS, and DANE if your DNS is signed

The two are complementary and neither substitutes for the other. Most organisations should have DMARC at enforcement before they start on this — not because MTA-STS is unimportant, but because impersonation is the more commonly exploited weakness.

Who this actually protects

Your correspondents as much as you. The policy you publish governs mail being sent to you, so what you are securing is other people's messages arriving at your servers — including everything they send you that you would rather nobody read.

The two pieces you publish

A DNS record and a file on a web server.

Two pieces, in two different places. One without the other does nothing.

1 — A DNS RECORD

`_mta-sts.yourdomain.com`
`v=STSV1; id=20260919a`

Announces that a policy exists, and changes when you update it.

2 — A FILE ON A WEB SERVER

`https://mta-sts.yourdomain.com/`
`.well-known/mta-sts.txt`

The policy itself: which servers are yours, and how strict to be.

The web server needs a valid HTTPS certificate for the mta-sts host. That certificate is what makes the policy trustworthy, so a self-signed one is no better than no policy.

Figure 2 — Both are required.

This is the part that surprises people: MTA-STS needs a web server. Not a big one — it serves a single text file — but it must answer on HTTPS at `mta-sts.yourdomain.com` with a certificate that a sending server will trust.

That certificate is doing real work. It is the reason a sending server can believe the policy came from you rather than from whoever is interfering with the connection.

The most common half-configuration

A DNS record published, and nothing serving the policy file. The domain looks configured to a casual glance and has no protection at all, because senders that cannot fetch a policy are required to deliver as though you had never published one.

What a sending server does

Four steps, then it remembers the answer.

What a sending server does, the first time it writes to you.

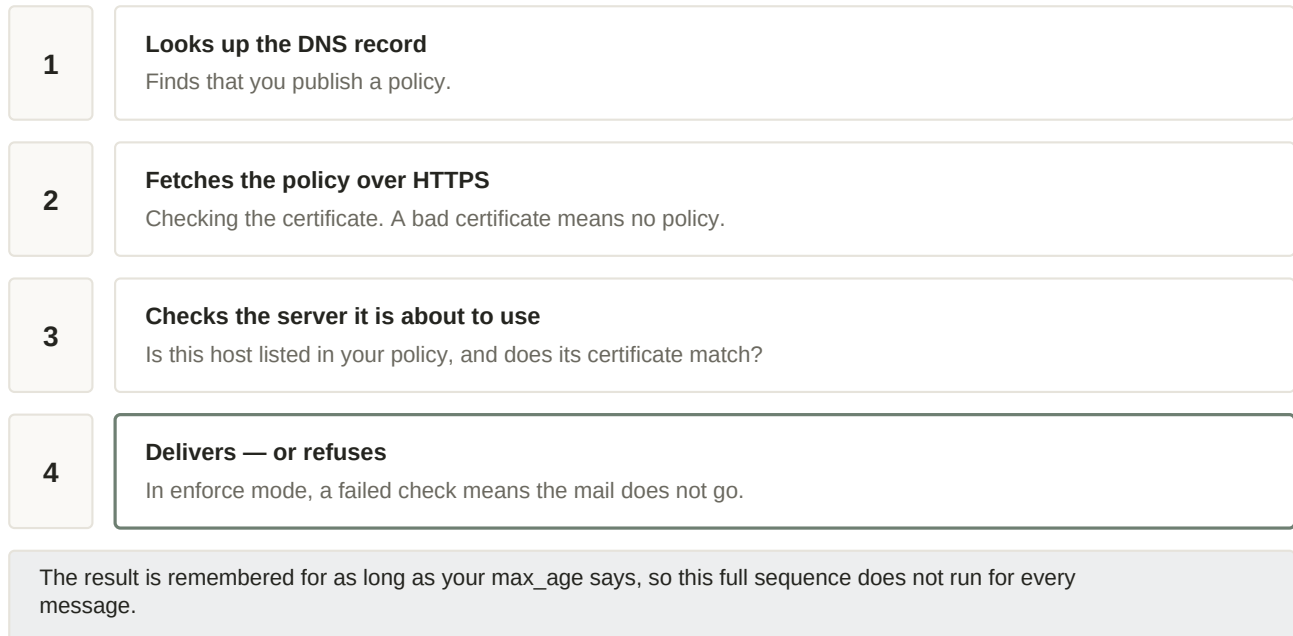


Figure 3 — Discovery, validation, delivery.

A few details of that sequence are worth knowing because they affect how you host the file. Sending servers will not follow redirects, so the policy must be served directly at that address rather than bounced to another one. They expect plain text, not HTML. And the response has to be a plain success — anything else is treated as no policy.

Failure is silent, by design

If your policy cannot be fetched, senders carry on as though you had never published it. Your mail keeps flowing and your protection quietly disappears. Nothing will alert you to this unless you are collecting reports — which is the subject of Section 09.

05

Reading the policy file

Four kinds of line, one of which you repeat.

served as plain text at /.well-known/mta-sts.txt

version: STSv1

The only version there is. Always this.

mode: enforce

How strict to be: enforce, testing or none. This is the setting that does the work.

mx: mail.example.com

A server allowed to receive your mail. One line per server, and every server you actually use must be listed.

mx: *.example.net

A wildcard covering one level only — see the next section.

max_age: 604800

How long senders may remember this policy, in seconds. Longer is safer; the maximum is 31557600, about a year.

One field per line. Nothing in it is secret — it is a public statement about your own servers.

Figure 4 — The policy file, pulled apart.

If you only ever check two, check mode and mx. The first decides whether the policy does anything. The second decides whether your own mail still arrives.

The three modes

Only one of them protects anything.

Three modes. Only one of them changes what happens to your mail.

none

Tells senders to forget any policy they have cached. Used to back out safely, not as a starting point.

Cancel the policy

testing

Failures are reported to you and the mail is delivered regardless. Nothing is protected — this is a rehearsal.

Report, but deliver anyway

enforce

If the connection cannot be secured to a server you listed, the mail is not delivered. This is the only mode that protects anything.

Refuse to send insecurely

A domain sitting in testing mode has the same transport security as a domain with no policy at all. The difference is that you now get told about it.

Figure 5 — none, testing, enforce.

The mode is also how you back out. If enforce turns out to be breaking delivery, setting mode: none and changing the id in your DNS record tells senders to discard what they cached. Without that, they would keep applying your old policy until max_age ran out.

Why max_age is a security setting

A cached policy keeps working even when an attacker is blocking every attempt to look up a new one. A short max_age means senders forget you quickly and have to re-discover the policy more often, and every re-discovery is another chance to interfere. Weeks or months is normal; a day is not.

Listing your servers correctly

The mistake that stops mail arriving.

Every server that receives your mail has to appear in the policy. Miss one, and in enforce mode senders will refuse to deliver to it. This is the single most likely way to break your own mail with MTA-STS, and it usually happens through a wildcard that covers less than expected.

A policy line of `mx: *.example.net` covers exactly this much:

<code>mail.example.net</code>	covered	one label replaced — covered
<code>example.net</code>	not	the wildcard needs a label to match
<code>foo.bar.example.net</code>	not	two labels deep — not covered

A wildcard replaces one label and no more. If a server of yours is not matched by any line, enforce mode will stop mail reaching it.

Figure 6 — A wildcard replaces one label, and only one.

- **Check your backup servers.** The ones that only take mail when the primary is down are the easiest to forget, and the failure appears at the worst possible moment.
- **Check what your provider actually uses.** Hosted mail often has more receiving hostnames than the obvious one, and they change.
- **List them all before you enforce.** In testing mode a missing server costs you a report; in enforce mode it costs you the mail.

The limitation worth knowing

MTA-STS protects senders who have met you before.

The one weakness worth understanding.

A sender who has never written to you before

has nothing cached. An attacker who can interfere with that very first policy lookup can make it appear that you publish no policy at all, and the sender will deliver as though you had not.

A sender who has written to you before

already holds your policy and will keep applying it for the whole of max_age, even if an attacker blocks every later lookup. This is why a long max_age is a security setting and not just a performance one.

In practice almost all of your mail comes from senders you have exchanged mail with before, which is what makes the protection worth having despite the gap.

Figure 7 — First contact is the gap.

This is a real weakness and it is worth being clear about it. MTA-STS builds its protection on a policy that has to be fetched at least once over the same internet an attacker may be interfering with. The first fetch is the moment of exposure.

DANE does not have this gap, because its trust comes from signed DNS rather than from a file that has to be retrieved. It needs a DNSSEC-signed zone, which is why MTA-STS exists at all — most domains do not have one.

	MTA-STS	DANE
What it trusts	Certificate authorities, as browsers do	DNSSEC signatures on your DNS
What you must run	A web server with a valid certificate	A DNSSEC-signed zone
If discovery is blocked	Falls back to no protection	Fails closed, if DNSSEC is validated

They are not rivals, and publishing both is common. If your DNS is not signed, DANE is not available to you and MTA-STS is the one to do.

Figure 8 — Two routes to the same goal, chosen by what your DNS already does.

Finding out whether it works

A second record, and the reason to publish it first.

TLS reporting is a companion standard. It is a separate DNS record asking senders to tell you, daily, how their connections to your servers went — how many succeeded, how many failed, and why.

```
_smtp._tls.yourdomain.com    TXT  
"v=TLSRPTv1; rua=mailto:tlsrpt@yourdomain.com"
```

Publish this *before* your MTA-STS policy, not after. It costs nothing, it cannot break anything, and it is the only way to find out what enforce mode would have done before you turn it on.

What the reports tell you

Which sending systems could not negotiate a secure connection to your servers, and whether the cause was an expired certificate, a hostname mismatch, or a server you forgot to list. Every one of those is mail that would stop arriving the day you move to enforce.

A safe order to do this in

Each step exists to make the next one boring.

- **1. Publish the TLS reporting record.** Nothing changes. Reports begin arriving.
- **2. Check your certificates.** Every server that receives your mail needs a valid, unexpired certificate matching its own hostname. This is where most problems are.
- **3. Serve the policy file in testing mode.** Still nothing changes for delivery, but senders now report against your actual policy rather than in the abstract.
- **4. Read the reports for a few weeks.** Fix every failure. Missing servers and mismatched certificates both show up here.
- **5. Switch to enforce, and raise max_age.** Only once the reports have been clean for a while.

The failure mode is undelivered mail

Unlike the authentication standards, a mistake here does not weaken your security — it stops messages arriving. That is why testing mode exists and why it is worth sitting in it longer than feels necessary.

A short checklist

What to publish, and what to watch for.

Getting it right

- **Do DMARC first** if you have not. Impersonation is the more commonly exploited problem.
- **Publish TLS reporting before anything else.** It is free and it cannot break delivery.
- **List every receiving server**, including backups, and remember a wildcard covers one label only.
- **Change the id whenever you change the policy.** Senders use it to notice the file has been updated.
- **Use a long max_age once settled.** Weeks or months. The maximum is about a year.

Signs something is wrong

- A DNS record exists but the policy file does not load. You have no protection and no warning.
- The policy is served over a redirect, or as HTML. Senders will reject both and treat you as having no policy.
- The certificate on `mta-sts.yourdomain.com` has expired. The policy stops being trusted the moment it does.
- You have been in testing mode for a year. That is the same transport security as publishing nothing.
- Reports show failures against a hostname you do not recognise. Find out what it is before you enforce.

Check your own domain

ReaperCloud's MTA-STS tool fetches your policy the way a sending server would — following the same rules about redirects, content type and certificates — and checks your listed servers against your real MX records. `reapercloud.com/mtasts.php`, or `reapercloud.com/report.php` for everything at once.

This document describes MTA-STS as specified in RFC 8461 and TLS reporting as specified in RFC 8460. It is an explanation, not a specification: where they differ, the RFCs are correct and this is simplified.