I figured out how DMARC works, and it almost broke me

How to prevent email spoofing on your domain, using an unholy combination of silly standards.

Recently, I encountered a problem. My domain didn't correctly implement SPF, DKIM, or DMARC.

Then, I encountered a second problem: I had no idea what those were, and seemingly *nobody* has written about SPF, DKIM, or DMARC in a way that a human can understand, not to mention implement. Every article I found was either highly technical, trying to game SEO to sell me something, or too high level to be useful.

As a result, I've had to do a lot of hard work and research to understand this problem. Hopefully, because I had to do this, you won't.

There's two main sections here: a human explanation of what these things are, followed by a reasonably straightforward way to implement them.

This might not be easy, but if you've landed here, it's probably not optional. I hope this helps.

I'm keeping this article up-to-date as I learn more, so it'll change and grow. If you want to keep track of what's changed since the last time you were here, check the diffs on GitHub.

Table of contents

What are these weird acronyms?

SPF, DKIM, and DMARC are complementary systems. SPF and DKIM are used by email servers as indicators of whether or not an email is spam. DMARC then does two things: it tells email servers how *important* SPF and DKIM are, *and* what to do when an email fails to pass their tests.

This probably doesn't make much sense yet - that's fine. Let's dig a little deeper.

SPF

SPF is a way to declare who's allowed to send emails from your domain. It stands for the "sender policy framework," but you don't need to know that. Just call it SPF, or "spoof." It's meant to make it harder to send spoof emails.

For example, it's a way to say "emails from mycompany.com can only be sent from Google and Postmark." Declaring SPF makes it harder for me to send emails from your domain in an attempt to phish.

Here's how it works, for a *valid, non-phishing* email:

1. I send an email to you from `hello@sadl.io`, using my Fastmail SMTP server.
2. Gmail (your email service) receives the email.
3. The email is from someone at `sadl.io`, so Gmail grabs the DNS records for `sadl.io`
4. `sadl.io` has a DNS record that declares its SPF policy. It says that emails can be sent from Fastmail.
5. This email was sent from Fastmail, so it passes the SPF test.
6. The email lands in your inbox.

That's all great! SPF hasn't stopped me from sending a real email to you. But it seems pretty simple. So... what *would* it stop?

Emails are notoriously easy to spoof. To me, even though I haven't written PHP in years, *nothing* demonstrates this more simply than this PHP script:

```php
<?php

// Adapted from this vulnerability report: https://hackerone.com/reports/34112

$to = "employee@example.com"; // Try to phish an employee of example.com
$headers = "From: payroll@example.com"; // Pretend I'm example.com's payroll account

$subject = "Verify your bank details for your paycheque"; // You can see where this is going...
$txt = "Hi Simon, we're updating our payroll software, and in order to continue receiving your paycheq

mail($to, $subject, $txt, $headers); // Yikes.
```

This script sends an email to `employee@example.com`, which looks like it comes from `payroll@example.com`, and asks the employee to enter their banking details. It's a pretty compelling email, and could probably get a few people to, at least, click the link - or worse.

You could run this script *right now*, and if `example.com` hasn't set up SPF, an email might actually reach the inbox of `employee@example.com`.

Seriously, it's that simple. We've all learned this at one time or another: emails are easy to spoof. It's always been just a cost of doing business. "You can't stop spoofers! Email is too complicated to fix!" are probably things you've heard, or said. We shove it to the back of our minds. We know emails are insecure, but we use them anyway. Turns out that's not entirely true. SPF can actually start to help cut down on spoofing.

If SPF were set up, here's what would happen when that script runs:

1. An email gets sent to `employee@example.com`.
2. This email gets received by Google, who run the mail server for `example.com`
3. Since it's an email from `example.com`, Google gets the DNS records for `example.com`
4. `example.com` has an SPF policy declared, which states that emails can only be sent from Google.
5. This email wasn't sent from Google; it was sent from a local mail server.
6. Since the sending domain doesn't match a domain that's allowed by `example.com`, the email is marked as spam. *Maybe*.

Wait... *maybe*? Yeah. SPF sounds great in principle. But it often has no effect without DMARC. We'll get to that, but suffice to say: if you've just set up SPF, it's mostly informational. Some email servers *might* use it, but they might not treat it with much importance. DMARC lets you increase its importance.

On envelopes vs. letters

This section gets even weirder, but bear with me. You might want to skim this now, read the rest of the article, and come back to it again after. It makes a lot more sense once you understand DMARC alignment, but we're not there yet. Alright. Here goes.

First off: shout out to Liam, who correctly pointed out an inaccuracy with this description of SPF. SPF *actually* checks the validity of the "envelope from", not "from".

Liam pointed me to an article that explains the difference between two things: an email (think of it as a letter), and the SMTP interactions that transport that letter (an envelope). It's from XEAMS, and you can (and should!) read it here. But I'll paraphrase it:

The **email** is what you see in your client. It includes the message body, and a whole bunch of headers. Those headers include the `From` header, which we spoofed in that PHP script, as well as the email subject, DKIM headers (we'll get to those in a moment!) and a whole bunch of other headers. Most email headers are soft-hidden from users by their email clients, but you can always view them - in your email client, whether it's an app or a website, you'll probably be able to find a menu item called "View original" or "View raw message". That'll show you what an email *really* is. But it doesn't show you the envelope.

The **envelope** is what happens at the SMTP level. I'll defer to the XEAMS article on what SMTP looks like, but in effect: the *envelope* can contain a *completely* different `MAIL FROM` email address than the email's `From` header.

You'll hear a few different terms thrown around to describe the SMTP `MAIL FROM` command: envelope from, return to, bounce address. They're all the same thing. I like "envelope from" colloquially, because the analogy is clever, but I think it's useful to know that it comes from the SMTP protocol command used to deliver your email.

Which brings us to this: SPF *doesn't actually care about the From header in your email*. It only cares about the `MAIL FROM` value.

Later on, we'll talk about DMARC alignment. But here's a spoiler: with DMARC alignment, DKIM notwithstanding, you're saying "in order for an email to be valid, the FROM address has to pass SPF, and the domain in the From *header* has to be the same as the domain in the FROM address." In effect, it makes that PHP spoofing example run *roughly* the way you'd expect, but it takes a bit of a circuitous route to get there.

I've left this inaccuracy in the above section. Why? Well... explaining the difference between envelope from and the From header has taken up as much space on this page as the explanation of SPF itself. If I explained this *first*, you'd be too far in the weeds by the time we get to the real example. Sorry for misleading you - but if it helps, I wrote the first version of this article not knowing the difference, and I still *essentially* got a decent understanding of SPF out of it.

DKIM

`DKIM` is a way to declare signing keys for emails from your domain. It stands for DomainKeys Identified Mail. Again... that doesn't matter. It's Dee-Kim. You can think of it as SPF's cryptographic cousin.

It means that an email server that receives an email can check if that email was sent by a server that knows a secret. Since it's public-key cryptography, there's two keys: a private one, held by your email sending server (SMTP) and known by no one else, and a public one, set in your DNS, which can be seen by *anybody* and used to determine if a signature was made using that secret key.

A domain can have multiple DKIM keys, by the way. That took me a long time to figure out. In all likelihood, most of the email sending services you use (Gmail, Office 365, Fastmail, Mailchimp, Postmark, Sendgrid, Mandrill, Postmark, or whatever) will provide a DKIM key you can add to your DNS records. If you add that key, you're authorising those services to send on behalf of your domain - it's similar to adding them in SPF, but instead of being about domains, it's about knowing a secret,

Here's how it works in a happy path:

1. I send an email from `hello@sadl.io` to `somebody@gmail.com`.
2. This email goes through my Fastmail SMTP server in order to be sent.
3. The Fastmail SMTP server generates a signature using the secret key, and attaches it to the email, then sends it to `example.com`'s receiving server.
4. The Google email server receives this email. It's from `sadl.io`, so it gets the DNS records.
5. The email has a signature embedded, and the DNS records for `sadl.io` declare a few public DKIM keys which can be used to verify that signature.
6. One of those DKIM keys matches the one used to make this signature - specifically, the one that Fastmail provides. So the DKIM test passes.
7. The email lands in `somebody@gmail.com`'s inbox.

The unhappy path here is fairly straightforward. You could use the same PHP script from the SPF section. Here's what would happen:

1. I run this script. An email gets sent to `employee@example.com`.

2. This email gets received by Google, who run the mail server for `example.com`
3. Since it's an email from `example.com`, Google gets the DNS records for `example.com`
4. The SPF test fails. But that *might* be ok, because `example.com` declares some DKIM keys in its DNS records.
5. This email doesn't have a signature attached. So none of those DKIM keys are useful. (Alternative: the email *does* have a signature attached, but it doesn't match the signature that would be generated by any of the public keys on `example.com`).
6. Since the email failed the SPF test *and* the DKIM test, it's marked as spam. *Maybe*.

Yup. Once again: *maybe*. An email could fail *both* of these tests, and still land in a user's inbox. You've put all this work in, and *still*, your email domain can be spoofed with 5 lines of *incredibly basic* PHP.

If you're me, it's taken hours to get to this point. You're dejected and demotivated. Everything you've tried has been hard to understand, and required a bunch of research. You've nervously updated DNS records, afraid that you're going to break your organisation's email services - only to discover that, on the contrary, you've effectively accomplished nothing. Everything is terrible. You briefly reconsider your career choices.

There's one acronym left on your list of things to implement: DMARC. Another miserable email security practice to implement, and - you expect - one more solution that leads to a dead end.

Alright. Fine. Screw it. Let's get into it. But if this doesn't work, I give up.

DMARC

I have good news for you. DMARC *will* make SPF and DKIM work better.

But there's some bad news, too: it's a bit complicated, and requires some more infrastructure.

So, what is DMARC? At its core, it's actually two things, packaged up into one DNS record:

1. It's a policy, which declares what email servers should do when an email fails SPF and DKIM tests
2. It's a reporting system, so that you can figure out who is trying to spoof your domain

You're probably more interested in 1, but 2 is super important. To understand why, let's talk about why SPF and DKIM don't do the thing you thought they did.

Emails are complicated. If you haven't come to that conclusion by this point, I'd like to talk to you. It's an old system that's held together by a bazillion standards that have been written over decades. People do unexpected things with emails. There are intricate systems that were set up in the 90s that still run, unobstructed, on top of email.

Changing how emails works tends to break things. And that's what SPF and DKIM do.

SPF and DKIM are used as *indicators* of whether an email is spoofed or not. But if you added an SPF record on your domain, and you forget to add one of your email systems - say, Postmark, which you use to send mission-critical notifications from your application to your customers - then your customers could stop getting emails. If you added DKIM keys to your domain, but one of your email services doesn't support DKIM - or you forgot to add DKIM keys for that service - your customers could stop getting emails.

But they *don't* stop getting emails. Some emails will fail both SPF *and* DKIM, and still end up in users' inboxes. It's entirely at the discretion of the receiving email server, and they can be quite lenient. That's good for the services you forgot to add, but it's bad for the spoofers you're trying to squash.

You want to do two things:

1. Find out what services you misconfigured, so you can fix them.
2. Stop spoofers from abusing your domain.

The "reporting" part of DMARC helps you, in the early days of your email domain security endeavour, figure out what services you misconfigured and need to update. You can enable the reporting part, *without* enforcing a strict SPF/DKIM policy. In other words: you can use DMARC to find out about emails sent from your domain that *would* fail the SPF/DKIM tests, *without* telling email servers that all emails failing those tests should be marked as spam.

Once you're satisfied that you've configured everything, and all the reports you get are for spoof emails, you can update your policy to tell email servers that those tests are now important. It gives you a bit of time to validate and test, before you switch over to enforcing your SPF/DKIM rules.

Implementing DMARC goes something like this:

1. Read this article, bash your head against a wall, openly weep.
2. Implement a lenient DMARC record in your domain's DNS, so that you start getting reports.
3. Read through the reports you get over the next days and weeks. Check if any valid emails are getting flagged for failing DKIM or SPF. If they are, fix them.
4. Update your DMARC record to make it less lenient.

Unfortunately, implementing DMARC is a process. When I started investigating this problem, I was hoping I could copy some good SPF, DKIM, and DMARC records, update them to match my domains, and implement them, all within an hour or two. That's not how it works. This takes some effort, and you need to let some time elapse.

In other words: email is complicated.

Luckily, as I said, I've done the hard work here, because I don't think *anybody* should have to figure this out on their own again. For you, this process will have to play out over the course of a few days or, more likely, a few weeks, but within a few minutes, you can be well on your way.

Let's talk about alignment

Here's the magic bit of DMARC, that makes all of this sing: it enforces *alignment* between either the *envelope from* (aka SMTP MAIL FROM) and the From header, *or* the domain on the DKIM signature and the From header. In both cases, the From header can be thought of as the anchor: it's the thing users see, so either SPF or DKIM needs to vouch that the From header is what it should be.

This finally does the thing we set out to do. That PHP script at the top will, now, only succeed in getting an email to the inbox if one of two things is true:

1. The PHP script is running on an IP address that is authorised to send for `example.com` using SPF, *and* both the SMTP MAIL FROM and the From header end in `example.com`, *or*
2. The PHP script is hooked up to a mail server that can sign an email with DKIM, using a key that's valid for `example.com`, and the From header ends in `example.com`.

(In both cases, assume the `mail()` function is hooked up to a mail server on the same IP. Otherwise, the words I have to use just become a jumble. The script could be hooked up to an external mail server, and as long as *that* mail server is allowed in SPF, or can sign for the domain with DKIM, it'll work fine.)

Now might be a good time to jump back to the envelope from section - it pairs nicely with this section.

How do I use them?

Here's the good news: you *probably* have SPF and DKIM records set up for your domain. These days, most email services will give you the correct DNS records to add, and offer tools to verify that they're set up correctly.

Implement SPF

SPF is set up as a TXT record in your domain's DNS. Here's what mine looks like for simonandrews.ca:

```
v=spf1 include:spf.messagingengine.com include:spf.mandrillapp.com -all
```

It's declared directly on `simonandrews.ca` as a TXT record - not on a subdomain. If I were sending emails from `newsletter.simonandrews.ca`, I'd need *another* record for that subdomain.

Most SPF records are going to look like this, though many will have more than one `include:` clause. Let's break this down:

- `v=spf1`: Declares this as an SPF record, as distinct from the other things you might declare in a TXT record. It's SPF v1.
- `include:spf.messagingengine.com`: Emails are allowed to be sent from anything `spf.messagingengine.com` (Fastmail's SPF subdomain) allows. Normally, this means that `spf.messagingengine.com` has its own SPF DNS record, which will probably list some valid IP addresses that emails can be sent from. There's also one for Mandrill, for transactional emails. You can have as many of these as you want. If you want to see what `include:` does after you've set up SPF on your domain, check out the dmarc analyser SPF Record Checker.
- `-all`: This one's confusing, so pay attention. You might think it's an argument, like you'd use on a terminal command. It's not. It's actually saying "fail" (`-`) on "all others" (`all`). In other words: "if none of the previous declarations matched an email you received from our domain, then the email you received is probably spoofed."

The last one, `-all`, is especially confusing because of what many email services recommend - which is `~all`. That's a tilde, not a dash. It's easy to miss. The tilde means "this should fail, but don't do anything." Some other services suggest `?all`, which may look like an encoding error, but actually means "if an email doesn't match these domains, it doesn't matter." *If you use `~all` or `?all`, your SPF record isn't doing very much, even if you set up DMARC*. Use `-all` instead. **Seriously, use `-all` instead.** Always put it at the end of your SPF record.

Like I said, most email services these days provide the SPF record you should set at the time you set up the service, and they will then *validate* your SPF record by checking your DNS from their side. You should double check with each of your providers that your SPF is currently configured the way it should be. In the service's dashboard or configuration, there's probably a "domains" section that will guide you and validate your settings. Their support teams may be able to help, failing that. And, remember: if they suggest `~all` or `?all`, ignore it and use `-all` instead.

N.B. SPF used to be its own DNS record type - instead of declaring a TXT record in your domain's DNS that contains SPF information, you'd declare an SPF record. If your domain has SPF-type records and TXT-type records, remove the SPF records. If you *only* have SPF-type records, move them to be TXT-type records. It's DNS, so this probably gives you anxiety, but trust me here. SPF-type records are going away, and all email services that obey SPF records will read them from TXT-type records. You want to do this.

Main takeaway

You probably have SPF set up, even if you didn't know you were setting it up at the time. It was probably just one of a hundred small tasks you did when you signed up for an emailing platform. But you should check it. Use your email services' tools to check them, but do a manual check, too. And never forget to use `-all`.

Resources

- dmarcian has a good overview of the syntax for SPF records. There's quite a lot more available than I've described, but in my experience, you don't need all that detail. Regardless, now that you know the basics (congratulations, by the way!), this page should make sense.
- The dmarc analyser SPF Record Checker is a good way to check your SPF records.

Implement DKIM

With DKIM, you're even more at the whims of your email providers. They might provide instructions to set up DKIM, like with SPF. If they don't provide instructions, it probably means they don't support DKIM. If they *do* provide instructions, you'll want to follow them *to the letter*.

Normally, a DKIM public key lives in your DNS records on a subdomain of `_domainkey.yourdomain.net` - for example, for Fastmail, I have keys on `fm1._domainkey.simonandrews.ca`, `fm2._domainkey.simonandrews.ca`, and `fm3._domainkey.simonandrews.ca`, per their instructions. It might be a TXT record, with the key directly inline, or a `CNAME` record that points to a public key hosted by your email providers.

When an email gets signed by the sending server using DKIM, the signature will include the identifier to use - for example, on my domain, it would be one of `fm1`, `fm2`, or `fm3`.

I won't get into the structure here *at all*. It varies by provider, but *will* end up with a DNS record underneath `_domainkey.yourdomain.net`. Again, there's probably something in a service's Domains dashboard that tells you what to do, and validates your configuration. You may have set DKIM up already, and didn't know it.

Main takeaway

Your email providers will tell you how to set up DKIM, if they support it. Trust them, and follow their instructions. If they don't support DKIM, it's not the end of the world - but make sure SPF is set up correctly.

Resources

- Postmark have a very good write-up of DKIM. It's better than anything I could write - though understanding the basics, as you do now, will help you understand that document on your first read.
- Once again, dmarc analyser have a good DKIM record checker you can use to validate your setup. You'll need your domain name, and the key (like `fm1` for Fastmail), because DKIM is declared on subdomains, not your domain root.

Implement DMARC

Oh boy, you're gonna hate this part.

This is the part that I've found most difficult to research. There are a *bunch* of companies that are gaming SEO to promote their DMARC analysis service - but none of them explain what DMARC is, at least not publicly. They don't give you the information to understand what's going on and what you'll need to do - instead, they're trying to sell you something that will just fix it for you.

I don't trust companies that game SEO, so I didn't trust any of these articles. I wanted to get down to basics.

Here's the bad news, though. I glossed over something earlier, when we talked about reporting. Forgive me. I didn't want to scare you away. We're *so close* now. You understand most of the concepts we're dealing with, and even if your email domains aren't locked down yet, they're definitely better. But... DMARC reports are sent as XML files.

I'm so, so sorry.

DMARC reports aren't meant to be read by humans. If you have a domain that has any volume of emails, you'll probably get a lot of DMARC reports of spoofing. Those reports are meant to be interpreted and aggregated by a machine.

In other words: unless you're willing to manually read XML reports to figure out what's happening (hey, on small domains, that's totally possible!), you're either going to be setting up some software on a server, or using a third party provider. Those companies that are gaming SEO, and being cryptic about what DMARC actually is so they can sell you something? You're probably going to give one of them some money.

Anyway. Let's start with the basics. Pretend that I didn't just say that. While the reports aren't fun to interpret manually, it'd be good to get our hands on some so we can get a feel for what this is. We're going to go through 3 phases:

1. Implement a lenient DMARC policy and start collecting reports
2. Wait, watch and analyze
3. Increase the strictness of your DMARC policy

That's not too bad. Let's start with a super simple DMARC policy.

The lenient policy

Set up an email address that can receive the reports. I used postmaster@simonandrews.ca, which actually just forwards to my personal email address. Then, I added a TXT-type DNS record on `_dmarc.simonandrews.ca` that looks like this - yours will go on `_dmarc.mydomain.net`:

```
v=DMARC1;p=none;pct=100;rua=mailto:postmaster@simonandrews.ca;ruf=mailto:postmaster@simonandrews.ca
```

Whoa whoa whoa... what's that? Ok. Let's break it down.

There's the *policy* declarations:

- `v=DMARC1` says "this is DMARC version 1"
- p=none says "if SPF or DKIM fail, you don't need to do anything." Remember when I said we'd implement a lenient policy? This is it.
- `pct=100` says "apply this policy to *all* emails from my domain"

And then we've got the *reporting* declarations:

```
rua=mailto:postmaster@simonandrews.ca
```

This tells an email server to send **aggregate** reports of spoofed emails to `postmaster@simonandrews.ca`

```
ruf=mailto:postmaster@simonandrews.ca
```

This tells an email server to send **forensic** reports of spoofed emails to `postmaster@simonandrews.ca`

This policy... works. If you set up your own postmaster email address, and use this DMARC record, you'll be well on your way. Go ahead and do that. Within a few days, you'll start getting reports.

Don't believe me? Try setting up DMARC using a configuration like this, giving it a bit of time to propagate (because DNS is slow), and then run the PHP script from earlier, using your domain and a "to" email address you own. You won't get a report right away, and the spoofed email generated by the script might even end up in your inbox. But within 24 hours, you'll probably get your first

DMARC report.

Sit back and watch

Start collecting these reports. If it's just a personal domain, you might be happy to look through them manually. If it's your company's domain, you might get a high volume of reports - you'll probably want to stand up a DMARC server like lafayette, or use a third party service, to handle these reports.

Regardless, after you've got some reports through, start analysing the reports. Are emails failing SPF or DKIM tests, that shouldn't be? Did you misconfigure a service? Are you or your company sending emails with a service you weren't aware of? Or, is there a spoofer who's aggressively abusing your domain?

What can you expect? Well, here's a report I got, after I ran that PHP script from earlier and sent to a Gmail address I have. I added in some comments, to outline what it's telling you.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<feedback>
  <report_metadata>
    <!-- Details who prepared this report (Google's mail servers) -->
    <org_name>google.com</org_name>
    <email>noreply-dmarc-support@google.com</email>
    <extra_contact_info>https://support.google.com/a/answer/2466580</extra_contact_info>
    <report_id>14538673265069095400</report_id>
    <date_range>
      <begin>1628899200</begin>
      <end>1628985599</end>
    </date_range>
  </report_metadata>
  <policy_published>
    <!-- This is the DMARC policy that Google's mail servers found for simonandrews.ca -->
    <domain>simonandrews.ca</domain>
    <adkim>r</adkim>
    <aspf>r</aspf>
    <p>none</p> <!-- This is what we set, p=none -->
    <sp>none</sp>
    <pct>100</pct> <!-- This is also what we set - pct=100 -->
  </policy_published>
  <record>
    <!-- Here's some details about the offending email(s) Google received -->
    <row>
      <source_ip>81.100.0.0</source_ip> <!-- (I censored out the IP) -->
      <count>2</count> <!-- I ran the script 2 times, which is reflected here -->
      <policy_evaluated>
        <dkim>fail</dkim> <!-- DKIM failed -->
        <spf>fail</spf> <!-- SPF failed -->
        <disposition>none</disposition> <!-- But, since p=none, no action was taken, except producing
      </policy_evaluated>
    </row>
    <identifiers>
      <header_from>simonandrews.ca</header_from> <!-- These emails were trying to spoof simonandrews.c
    </identifiers>
    <auth_results>
      <spf>
        <domain>simons-mbp.lan</domain> <!-- But they were actually sent from simons-mbp.lan -->
        <result>none</result>
      </spf>
    </auth_results>
  </record>
</feedback>
```

If you're running a large domain, you'll get a bunch of these reports. If you're running a small one, you might be able to handle it yourself.

Regardless, with a bit of time, you'll be confident you've set the policy correctly.

Increase the strictness

When we first added DMARC, we set p=none. That says, basically, "don't *do* anything, just tell me about SPF and DKIM when they fail."

If you're comfortable with the reports you've been receiving - that is, all of the emails in the reports are from spoofers, and all of the emails *you're* sending are passing the tests - you can increase this parameter. You have two options:

1. Set `p=quarantine`. This tells the receiving server that, if both SPF and DKIM fail, the email should be quarantined. That might mean it gets marked as spam right away, and sent to the receiver's spam folder. It might mean that it's put in a quarantine where an administrator for the receiving domain can approve or reject it. Either way, this option is good: it locks your domain down, but gives users recourse when something goes wrong.
2. Set `p=reject`. This is stricter. It tells the receiving server that, if both SPF and DKIM fail, the email should just be rejected. If you're *absolutely confident* that you've got everything configured correctly, *and* you're confident that future services will be set up correctly, then this is probably the option for you. I'd say it's fine to set `p=quarantine`, and then ramp yourself up to set `p=reject` at a later date.

It's in your hands now, though.

Resources

- dmarc.org have a good overview of what DMARC records are made of. If you want to skip right to the meaty stuff, read the section titled "Anatomy of a DMARC resource record in the DNS"
- dmarc.org also have a list of code and libraries you can use. That's where I found lafayette.
- dmarcian have a DMARC Record Checker, which will help validate that your policy is set up well.
- Postmark have a guide to DMARC tools. This guide is a good, fairly-unbiased breakdown. You're likely going to pay for a service here, ~~and Postmark come at it with an unbiased perspective because - as of this writing - they don't provide such a service themselves.~~ but Postmark give a reasonable breakdown (it turns out DMARC Digests is a service they provide, so it's not an unbiased source. Thanks to max1cc on Hacker News for pointing this out.)

Troubleshooting

Not receiving reports? You might need EDV

Thanks to Nicolas Lenz for reaching out with this!

If you've set everything up right, and are waiting for DMARC reports that just aren't coming, you might need to set up EDV.

You can't send DMARC reports to *just anyone*. I can't, for example, set up DMARC on `simonandrews.ca` and send the reports to `tim@apple.com`. It's likely that Tim Cook doesn't want to know about the deliverability of mail from my domain (his loss). To prevent annoying, unsolicited reports, DMARC implements a system known as EDV (which, depending on who you talk to, means either External Domain Verification or External Destination Verification).

So, when might you need it? You'd need to meet these two main conditions:

1. You're sending your DMARC reports to a different domain. (EDV wouldn't be necessary if I were sending reports from `simonandrews.ca` to `postmaster@simonandrews.ca`, because it's not external. Remember: *external* domain verification.)
2. You're *not* using a third-party DMARC reporting service. (These tools should have EDV set up automatically.)

An example that *would* require EDV: I'm setting up DMARC on `sadl.io`, a domain that I own, and I want to send reports to postmaster@simonandrews.ca, an email address on another domain that I own. In order for reporting to work, I'd need to configure DMARC on `sadl.io`, *and* configure EDV on `simonandrews.ca` to allow DMARC reports related to `sadl.io`.

Once you understand all of that, it's easy: add a DNS TXT record on the receiving domain (`simonandrews.ca`, in this case), on the `sadl.io._report._dmarc` subdomain, with a value of `v=DMARC1`. In other words, something like...

```
sadl.io._report._dmarc.simonandrews.ca TXT "v=DMARC1"
```

Now, before a provider sends off a report to `postmaster@simonandrews.ca`, for emails related to `sadl.io`, it'll check `sadl.io._report._dmarc.simonandrews.ca` to see if it's willing to receive them.

You can find more on the dmarc.org article, Receiving DMARC Reports Outside Your Domain.

Wrapping up

Chances are, you should implement all of these. But your implementation will probably fork off from mine, as you learn about it.

My main goal here was to provide you with a vocabulary, so you understand what SPF, DKIM, and DMARC are. I'm not an expert. I probably got some things wrong here. And this article, explicitly, only goes into the basics.

But hopefully, now that you're here, you've got a basic understanding, and perhaps the start of an implementation. You can go read some other posts, and you won't be completely lost. I'd suggest looking at some of the resources I linked - they're not too complicated, and most won't try to sell you things.

Thanks for coming along on this crazy, not-really-that-satisfying journey with me. I'll try and keep this post updated as I learn more, and welcome any and all feedback you might have. If I got anything wrong here (I'm absolutely certain I did), please reach out to hello@sadl.io.

Corrections

You can take a look through all the changes I've made on GitHub - if you've read this article before, and just want to check the new stuff, I'd suggest reading the diffs.

1. A previous version of this article claimed that Postmark provided an unbiased view on DMARC analysis services, because they did not provide such a service. In fact, they do, called DMARC Digests, which is mentioned in their guide. **Thanks to max1cc on Hacker News for correcting this.**
2. The SPF section claims that SPF works on the From header. In fact, it works on the "envelope from", which only exists at the SMTP transport level. Rather than correcting this (I *did* try, but the section came out too complex), I added an additional section that explains the difference. **Thanks to Liam for reaching out over email and setting me straight.**

Social

- simon360 on GitHub
- simon360 on Instagram
- sadl-uk on LinkedIn
- @simon360 on Twitter

Contact

You can get in touch with me at hello@sadl.io, or via my social media profiles. I am not currently looking for new work.

Top of page ↑