Proofpoint Email Authentication Guide

This guide provides step-by-step instructions for configuring SPF, DKIM, and DMARC on Proofpoint cloud email gateways.

Note: This guide does not apply to Proofpoint Essentials, which does not currently support DKIM or DMARC.

Enable SPF checks for inbound mail

SPF is one of the authentication mechanisms used by DMARC. SPF checks should already be enabled by default in your Proofpoint instance. To check this, navigate to **Email Protection> Email Authentication> SPF> General**, and ensure that:

1. **Enable** is set to **on**
2. **Restrict processing to selected policy routes…** is checked
3. Only the **default_inbound** policy route is in the **Require Any Of** list

Then click **Save Changes**.

Publishing a SPF record

Most domains already have a SPF record published in DNS. You can use the MX Toolbox SPF Record Tool to check this.

If you don't already have a SPF record published in your domain's DNS, add the following basic SPF record as a TXT record at the root of your domain's DNS zone:

"v=spf1 mx ~all"

This will explicitly allow email that comes directly from the gateways listed in your domain's MX records. Additional record modifications to the SPF are required if you send email from sources other than these gateways.

For more information on building a SPF record, see this guide.

DKIM signing

Along with SPF, DKIM is one of email authentication mechanisms used by DMARC. DKIM is more reliable than SPF, because unlike SPF, it survives when messages are automatically forwarded. You must configure outbound email to be DKIM signed.

If you use Proofpoint as your as your outbound email gateway, you can configure DKIM signing by completing the following steps:

Navigate to **Email Protection> Email Authentication> DKIM Signing> Keys**.

For **each** email domain**:**

1. Click **Generate Key**
2. In the **Domain** field, enter the **base** domain name, such as example.org, even if your email addresses are john.smith@hq.example.org
3. Enter an arbitrary string in the selector field. Something simple such as **s1** will do. **Note**: The DKIM selector is visible in all signed email headers.
4. Leave the **Scope** set to **Any**
5. Click **Add Entry**
6. After the DKIM Key table row populates (which can take a few minutes), click on the **Show** button in the **DNS Text Record** field
7. Publish the displayed record in the public DNS zone for your domain

Navigate to **Email Protection> Email Authentication> DKIM Signing> General**.

Set **Enable** to **On**

Click **Save Changes**.

Create email authentication quarantine folders

Navigate to **System> Quarantine > Folders**, and add two new folders

Email authentication failures

We'll configure Proofpoint to copy emails here that fail DMARC with a message from header from domain that has a DMARC policy of quarantine or reject. That way, it is easy to see which emails are being blocked by DMARC.This is useful for gathering threat intelligence, and for potential troubleshooting with domain owners who have prematurely set an enforced DMARC policy.

Set the quarantine folder disposition to delete messages after **30 days**.

Email authentication failures from our domains

We'll configure Proofpoint copy emails here if they fail DMARC with a message from header with a domain that matches one of our domains. This makes it easy to view the DMARC failures from only the domains that we control.

Another key difference is, we will configure Proofpoint to copy messages here even if the DMARC policy is set to monitor only (p=none). That way, email sources that are failing DMARC can be identified and fixed before moving the the domain to an enforced DMARC policy (p=quarantine or p=reject). This is very useful, as Proofpoint does not currently send aggregate DMARC reports.

Set the quarantine folder disposition to delete messages after **30 days**.

Create a new Policy Route

Navage to **System> Policy Route**, and add a new Policy Route.

Set the **Route ID** to **from_our_domains**.

Set the **Description** to **Emails with a message header from domain matching one of our domains**.

Add a new condition to this Policy Route for **each** of your domains:

Condition: **Message Header From (Address Only)**
Operator: **Ends With**
Value: **@example.com**

Where example.com is your domain name.

Warning: It is important to use the **Ends With** operator, and start the value with **@**. Otherwise, the condition could match badexample.com or example.commnity.

Configure the Proofpoint DMARC policies

As mentioned in the Create email authentication quarantine folders section, we will configure Proofpoint to use the Policy Route we just created to treat email from our domains slightly differently than email from other domains.

Create a new from_our_domains DMARC policy

Navigate to **Email Protection> Email Authentication> DMARC> Policies**.

Create a new DMARC policy called **from_our_domains** with a **Description** of **Emails with a message header from domain matching one of our domains**.

Edit the default DMARC policy rules

Navigate to **Email Protection> Email Authentication> DMARC> Rules**.

Select the **default** DMARC policy.

Edit the **quarantine** rule.

Set the quarantine folder to **Email authentication failures**.

Set the **Delivery Method** to **Discard**.

Click **Save Changes**.

Edit the **reject** rule.

Check **Quarantine message…** and select the **Email authentication failures** folder.

Set the **Delivery Method** to **Reject**.

Click **Save Changes**.

Edit the from_our_domains DMARC policy rules

Navigate to **Email Protection> Email Authentication> DMARC> Rules**.

Select the **from_our_domains** DMARC policy.

Check **Restrict processing to selected policy routes…**

Move the **from_our_domains** Policy Route from the **Available** list to the **Require Any Of** list

Edit the **quarantine** rule.

Set the quarantine folder to **Email authentication failures**.

Set the **Delivery Method** to **Discard**.

Click **Save Changes**.

Edit the **reject** rule.

Check **Quarantine message…** and select the **Email authentication failures** folder.

Set the **Delivery Method** to **Reject**.

Click **Save Changes**.

Edit the **none** rule.

Check **Quarantine message…** and select the **Email authentication failures** folder.

Set the **Delivery Method** to **Continue**.

Click **Save Changes**.

Enable DKIM checks for inbound mail

navigate to **Email Protection> Email Authentication> DKIM> General**, and ensure that:

4. **Enable** is set to **on**

5. **Restrict processing to selected policy routes…** is checked
6. Only the **default_inbound** policy route is in the **Require Any Of** list

Then click **Save Changes**.

Enable DMARC checks for inbound mail

navigate to **Email Protection> Email Authentication> DMARC> General**, and ensure that:

7. **Enable** is set to **on**
8. **Restrict processing to selected policy routes…** is checked
9. Only the **default_inbound** policy route is in the **Require Any Of** list

Then click **Save Changes**.

Troubleshooting

When reviewing emails in the quarantine folders or in your own inbox, look for the Authentication-Results message header from ppops.net. For example:

```
Authentication-Results: ppops.net;
        spf=pass smtp.mailfrom=example@gmail.com;
        dkim=pass header.s=20161025 header.d=gmail.com;
        dmarc=pass header.from=gmail.com
```

This header tells you how Proofpoint evaluated the message’s authenticity using SPF, DKIM, and DMARC.

In the example above, we can determine that:

- Proofpoint looked at SPF record at the domain in the mail from SMTP mail from header, and checked the SPF record at gmail.com. SPF passed.
- Proofpoint checked the DKIM signature using the public key at 20161025._domainkey.gmail.com, and the signature was valid.
- DMARC passed because **at least one** of these conditions were true:
    - SPF passed **and** the SMTP mail from domain (which SPF uses to decide which domain’s SPF record to check) matches the message from header base domain
    - The DKIM signature passes **and** the base domain of the public key (the d value) matches the base domain in the message from header.