Authentication for Email Senders

Florian Vierke – florian.vierke@mapp.com · Sebastiaan de Vos – sebastiaan@inboxsys.com · Michael Kliewe – m.kliewe@team.mail.de – Version 0.8, 15.03.2023

Table of contents

Document history

Title	Title	Date	Authored by
Initials	{doctitle}	15.03.2023	Florian Vierke, Sebastiaan de Vos, Michael Kliewe

Version	Date	Description	Initials
0.7	02.12.2022	Update	SdV, MK
0.6	23.11.2022	Small changes	FV
0.5	11.11.2022	Structural change	SdV
0.4	07.11.2022	Evaluating DMARC reports	SdV
0.3	28.09.2022	Additions to DMARC, Sections 4.1 and 4.2	MK
0.2	28.09.2022	Sections 3 and 4 added as draft	FV
0.1	20.07.2022	First draft	FV

Forging sender addresses to take on a false identity is one of the most common forms of email fraud. Attackers aim to elicit information from their victim (e.g. phishing) or persuade them to commit an act useful to the attackers (e.g. CEO fraud) by pretending to be someone else. This leads victims to distrust email in general and can cause significant economic damage to both individuals and businesses. In recent years, email experts have developed several methods to curb this form of abuse.

This document looks at email authentication from the perspective of a sending mail system. It gives configuration examples for SPF, DKIM and DMARC so that email can be authenticated before receipt, identity abuse detected, and recipients protected from abusive messages. We also cover software examples that help to evaluate DMARC reports. The aim is to only send messages that comply with the sender-side guidelines for SPF, DKIM and DMARC.

Email authentication for receivers?

It is equally important to check the origin/authentication of incoming emails and to send DMARC reports for incoming emails.

However, this document does not cover what needs to be done to achieve this. Information on this can be found at https://www.eco.de/themen/e-mail/downloads/email-authentication/. This document is specifically aimed at receivers.

The following sections show you how to use SPF, DKIM and DMARC to authenticate the emails you send.

Terminologies

Letter	Email Part	Designation according to RFC	Designation in this document
Sender on the envelope	Message Envelope	RFC5321.MailFrom	Envelope Sender
Receiver on the envelope	Message Envelope	RFC5321.RcptTo	Receiver
Sender on letter	Message Header	RFC5322.From	From Header

1. Why authenticate?

A basic prerequisite for being able to communicate meaningfully at all (be it by email or via another channel) is that the two communication partners are known and trustworthy. In the case of email, it is crucial whether we know the sender – and can verify that they are indeed the communication partner they claim to be. Without successful verification, the content of the message is worthless, possibly even dangerous.

Combating spam is a central challenge for mailbox providers. For this reason, more and more mailbox providers on the receiving side are moving to only accept (bulk) emails from authenticated sending domains.

1.1. Why authenticate with DMARC?

To prevent misuse of your own sending domain, it is important to understand the basic idea of DMARC: DMARC publishes guidelines for handling breaches of SPF and DKIM. DMARC requires that an email conforms to at least one of the two methods SPF or DMARC. If both methods fail, an email is considered to be not authentic.

If an attacker misuses a foreign domain for illegitimate message sending, this can negatively impact deliverability. DMARC serves to prevent misuse.

This is where the policy comes in, which DMARC publishes with the help of the `p-tag` in the DMARC entry in the DNS of the `From:` header domain. Three values are permitted for the `p-tag`:

`none`

If `none` is set, the sender domain specified in the `From:` header requests that no action be taken when SPF and DKIM violations occur.

`quarantine`

If `quarantine` is set, the sender domain specified in the `From:` header requests that the message be accepted but not delivered directly to the inbox, but placed in quarantine, e.g. the spam folder.

`reject`

If `reject` is set, the sender domain specified in the `From:` header requests that acceptance of the message be refused and that the message not be delivered.

DMARC is only fully activated with a "reject" policy. This allows receivers to consider domain reputation in addition to IP reputation. Thus, deliverability may be less affected due to the – often shared – IP reputation.

BIMI

Brand Indicators for Message Identification (BIMI) makes it possible to link a brand logo and a certificate at a predefined point in the DNS. Mailbox providers on the receiving side can integrate this logo in email programs or webmail interfaces and display it as the sender logo.

BIMI requires DMARC to have a "reject" policy on the organisational domain.

2. Implementation

The three main methods are used in combination to a) legitimise the sending systems for an envelope sender domain (SPF), b) verify the identity of a domain (DKIM) and c) to set a policy (DMARC) on how to deal with messages that do not comply with SPF and DKIM, as well as to receive reports on the current status of possible identity abuse. These three methods can be grouped under the term "Email Authentication".

Email Authentication

In its original form, the medium of email does not contain any possibility of authentication. We only see the IP address of the last, forwarding server. However, this does not necessarily have to be the sending mail server. There are no native control options for domains.

"Email Authentication" combines the SPF, DKIM and DMARC methods into a mechanism that can be used to check incoming emails for authenticity. Seen in isolation, the methods provide the following possibilities:

2.1. SPF

SPF is used to store the IPs that are allowed to send for the respective dispatch domain in a DNS TXT entry. Setting an SPF DNS record is simple and should be implemented by senders.

All IP addresses or IP ranges that are used for sending are required. These are recorded in an SPF entry separated by spaces. The finished entry might look something like this:

`DOMAIN.TLD TXT "v=spf1 ip4=192.0.2.0 ip4=192.1.2.0/24 ip6=fe80::0202:b3ff:fe1e:8329/64 include:sub.example.com -all"`

A redirect can also be included:

`DOMAIN.TLD TXT "v=spf1 redirect:sub.example.com"`

If a redirect is set, no other parameters, apart from "v", are allowed. E.g. -all or ~all should also be omitted! The latter is a common mistake.

SPF version

There is only one version of SPF: spf1. The version "spf2.0/*" is not actually an SPF version, but rather a SenderID and now obsolete. More information here: http://www.open-spf.org/SPF_vs_Sender_ID.

The above-mentioned examples are just examples. SPF is more complicated than that! For further details on the implementation, please refer to https://www.rfc-editor.org/rfc/rfc7208.

Unfortunately, SPF fails short in many core use cases of email, such as forwarding or using mailing lists. Therefore, SPF is almost exclusively used in combination with other authentication methods.

2.2. DKIM

The implementation of **DKIM** is, therefore, much more important. With this authentication method, the email is signed with a private key and the matching public key is stored in the DNS. Receivers can now check whether the signature of the incoming email is valid. If this is the case, then it confirms that:

1. The sender holds the private key.
2. The sender has access to the DNS of the signing domain.
3. The content of the received email (at least the signed parts) has not been changed in transit by any forwarding mail server.

If *aligned is* signed, i.e. the domain in the `From:` header of the email and the DKIM domain belong to the same organisational domain, we can thus ensure that the sender of the email was actually authorised by the domain owner to send it.

In this way, not only the integrity but also the authenticity can be determined.

The public part of the key is stored in the DNS of the signing domain. A selector is necessary for this. This is what a DKIM key looks like in the DNS:

`SELECTOR._domainkey.DOMAIN.TLD TXT "v=DKIM1; k=rsa; p=PUBLIC_KEY"`

This makes it possible to set several DKIM keys in a domain – using different selectors. Rotating keys and selectors on a regular basis and, in larger organisations, using different key/selector pairs for different purposes is also recommended.

The DKIM should also be signed_(aligned_) with the sending domain in the FROM: header, as this is a requirement of DMARC.

Apart from the v (version), k (encryption type) and p (key) parameters, there are other parameters that can be set. Currently, only the "DKIM1" version is available. In addition to RSA, ED25519 can also be used for encryption. For further details on the implementation, please refer to https://www.dkim.org.

DKIM signatures can be implemented with various solutions, for example OpenDKIM or rspamd.

2.3. DMARC

DMARC requires successful authentication via SPF or DKIM of the `From:` header domain for a message. In addition, DMARC specifies which policy should be applied in the event of SPF and DKIM violations **and** enables the receipt of so-called feedback reports on the authentication results of a domain by storing a contact address.

For this, a DNS TXT record must be found under _dmarc.<domain>, which looks something like this:

`_dmarc.DOMAIN.TLD TXT "v=DMARC1; p=reject; rua=mailto:<reporting-address>"`

As a rule, daily DMARC reports in XML format are now received at <reporting-address> from participating mailbox providers. These should be evaluated graphically. A description of how to do this is below.

Failure Reports

Besides "rua", a "ruf" parameter is also allowed. A reporting address is also given here. Ad-hoc error reports are sent to this reporting address. The use of the RUF parameter is controversial in Europe for data protection reasons, as Failure Reports contain the entire email including subject and content.

Common mailing list errors leading to DMARC problems:

- Keeping the From: header,
- Adding the list name "[list name]" in the subject line,
- Adding footer text in the body text, or
- Adding a Reply-To: header that was "non-existent" due to oversigning.

Before you can use p=reject, you should test the setup first. To do this, start with a record that is de facto ineffective:

`_dmarc.DOMAIN.TLD TXT "v=DMARC1; p=none; rua=mailto:<reporting-address>"`

While you get reports with p=none, the domain is not protected with DMARC.

First, the incoming DMARC reports should be reviewed over a period of 2-3 weeks. As soon as a sender is certain that only valid authenticated mailings are being sent, the p=reject policy should then be adopted to prevent abusive mailings from third parties via the sender's own domain.

The above-mentioned examples are just examples. Many other parameters are possible beyond the ones mentioned above. For further details on implementation, please refer to https://www.rfc-editor.org/rfc/rfc7489.

2.4. Receiving DMARC reports

If the reporting domain (RUA/RUF) and the sending domain are different, the RUA domain MUST be authenticated to receive reports for that domain using an additional subdomain (TXT _report._dmarc.senderdomain.com). Verifying External Destinations (https://tools.ietf.org/html/rfc7489#section-7.1) in RFC 7489 addresses this in detail.

The DMARC DNS record can be checked using online check tools, for example:

- mimecast DMARC Record Check (https://www.dmarcanalyzer.com/de/dmarc-de/dmarc-record-check/)
- dmarcian DMARC Record Checker (https://dmarcian.com/dmarc-inspector/)
- MxToolbox DMARC Check Tool (https://mxtoolbox.com/dmarc.aspx)
- InboxSys Domainchecker (https://app.inboxsys.com/domainchecker.php)

The mailbox that receives DMARC reports should:

- Be large enough for the expected volume of emails.
- Have the reception rate limit high enough to receive even larger amounts of emails per minute – even around midnight.
- Have some spam checks disabled, because DMARC reports contain .xml attachments with IP addresses that could be on blacklists.
- Allow attachments of type .gz or .xml.

There should be a regular check of whether the mailbox exists/is active and receives emails, because it is quite annoying for senders of DMARC reports if a target address is not reachable and bounces occur because, for example, the mailbox is full, no .gz attachments are accepted, emails are rejected because of spam classification, etc.

2.5. Evaluating DMARC reports

The received DMARC reports need to be evaluated. This evaluation should be automated. There are various commercial tools for this, e.g. Agari, DMARCIAN or DMARCadvisor, but also some free open source programmes that can be helpful. We will take a closer look at some of these free options here:

2.5.1. ParseDMARC

ParseDMARC is a small Python module that can import DMARC reports from an IMAP mailbox into an Elasticsearch database. Once imported, the DMARC results can be viewed on a Kibana or Splunk dashboard. Detailed documentation can be found at https://domainaware.github.io/parsedmarc/. There is also a ready-made Docker stack with parsedmarc, Elasticsearch and Kibana (https://github.com/patschi/parsedmarc-dockerized) that can be used for a quick test.

2.5.2. DMARC Viewer

DMARC Viewer is based on Django and Python and it imports DMARC reports from an email inbox into a Postgres database. This tool has an integrated web interface. The documentation for this tool can be found at https://github.com/dmarc-viewer/dmarc-viewer/.

2.5.3. DMARC Report

DMARC Report contains a parser in Python on the one hand and a viewer in PHP on the other. It is based on John Levine's rddmarc script (https://www.taugh.com/rddmarc/). Documentation and further links can be found here: https://www.techsneeze.com/dmarc-report/

2.6. Interpretation of the evaluation

In these DMARC evaluation tools, you not only see the emails you have sent yourself, but also which third parties have sent with this domain. These can be mailing lists or forwards, but also misuse of the domain.

There are various criteria according to which you can filter. For example, you can filter for IPs where DMARC failed. These can be your own IPs or those of others. You can filter further from here, e.g. for alignment errors, missing signatures and SPF fails.