Process DMARC reports with sieve

Marcel van der Boom

I get a lot of DMARC reports because I host mail for a couple of domains. Most of these mails require no attention as they are just notifications that others use one of our domains. I want to separate these mails from my normal mail workflow and auto archive them if I haven't looked at them within, say, 2 weeks.

Doing this with sieve server-side has my preference, but apparently it's not trivial to determine the age of a message, which is the core logic needed here. Also, the processing of sieve rules is normally only during reception of messages, not ad-hoc or on some other event, although dovecot and pigeonhole have some options for this, among others the sieve-filter tool.

I really only found one implemenation online which roughly solves the same problem I was having, but this involved more than needed I think.

My solution consists of 3 parts:

1. the sieve script that handles DMARC reports on reception and age-ing;
2. use of an extension that calls an external program to evaluate expressions to determine age;
3. a daily job that runs the sieve script in the scope of the designated folder.

Here's the sieve script which deals with DMARC reports both in the normal INBOX flow and a special treatment after 14 days. The latter part is not automatic by dovecot on reception of emails, but triggered by a run of the sieve-filter program.

```
require ["date","fileinto","relational","variables","environment","imap4flags",
         "vnd.dovecot.execute", "vnd.dovecot.environment"];

# Parameters
set "dmarc_folder" "Folder.for.dmarc-reports";
set "purge_days" "14";

# Move DMARC notifications when received
if environment :is "vnd.dovecot.default-mailbox" "INBOX" {
  if anyof (
    header :contains "From" "dmarcreport@microsoft.com",
    header :contains "From" "noreply-dmarc-support@google.com",
    header :contains "From" "opendmarc@mail.arctype.co",
    header :contains "From" "opendmarc@box.euandre.org"  )
  {
    addflag "\\Seen";
    fileinto "${dmarc_folder}";
    stop;
  }
}

# When running in the dmarc_folder, archive when age is <purge_days>
if environment :is "vnd.dovecot.default-mailbox" "${dmarc_folder}"
{
  if currentdate :matches "julian" "*"
  {
    # Run a simple bc expresssion to get <purge_days> ago from todays julian day
    execute :output "purge_date" "bc" "${1} - ${purge_days}";

    # Compare this with Date header and archive when age reached
    if date :value "le" "Date" "julian" "${purge_date}"
    {
      fileinto "Trash";
      stop;
    }
  }
}
```

The first part of the sieve script just moves the mails into the `dmarc-reports` folder and is a normal sieve processing rule. The second part runs if the default folder is the `dmarc-reports` folder. If so, it uses the `ext_program` extension of the sieve interpreter to let the `bc` program evaluate the expression for the age of the message.

This uses a tiny script in the configured sieve execute bin directory of the `ext_programs` extension

```
#!/bin/sh
echo ${1} | /usr/bin/bc
```

which just pipes the input given by the sieve line into the `bc` program. On returning, stdout is put into the `purge_date` variable. I'm using `execute` because I do not need to pipe the whole message into the external program, but specify input specifically.

With the above configuration I can set a cron job in the crontab of the `vmail` user to run

```
sieve-filter -We -u <mymailaccount> \
             /path/to/vmail/mymailaccount/sieve/dmarc-archiver.sieve \
             Folder.for.dmarc-reports
```

which executes the sieve script mentioned above in the IMAP folder `<dmarc_folder>` only.

I'm not sure why sieve makes it so difficult to get the age of an email (unless I'm missing something). Protonmail solves this by having a custom extension 'vnd.proton.eval' which does something similar like the above, but in the scope of the sieve language itself without having to shell out to an external program explicitly. (I think; I have not seen their implementation)

My approach above obviously has some drawbacks:

- the `bc` external program is called for every mail that matches, fine for 10 or 20 I guess, but rather inefficient if the amount of matched messages is big. For now, not a problem.
- unsure what sort of security consequences this has, the execution scope and environment is very limited, but we're still giving control to a script calling other programs.