By Falko Timme
Published: 2007-02-22 21:33

How To Implement SPF In Postfix

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 02/09/2007

This tutorial shows how to implement SPF (Sender Policy Framework) in a Postfix 2.x installation. The Sender Policy Framework is an open standard specifying a technical method to prevent sender address forgery (see **http://www.openspf.org/Introduction**). There are lots of SPF extensions and patches available for Postfix, but most require that you recompile Postfix. Therefore we will install the *postfix-policyd-spf-perl* package from **openspf.org** which is a Perl package and can be implemented in existing Postfix installations (no Postfix compilation required).

I want to say first that this is not the only way of setting up such a system. There are many ways of achieving this goal but this is the way I take. I do not issue any guarantee that this will work for you!

1 Preliminary Note

I assume that you have already set up a working Postfix mail server.

The following procedure is distribution-independent, i.e., it should work on any Linux distribution (however, I tested this on Debian Etch).

2 Install Required Perl Modules

The *postfix-policyd-spf-perl* package depends on the *Mail::SPF* and the *NetAddr::IP* Perl modules. Therefore we are going to install them now using the Perl shell. Start the Perl shell like this:

```
perl -MCPAN -e shell
```

If you run the Perl shell for the first time, you will be asked a few questions. You can accept all default values. You will also be asked about the CPAN repositories to use. Select repositories that are close to you.

After the initial Perl shell configuration, we can start to install the needed modules. To install *Mail::SPF*, simply run

```
install Mail::SPF
```

In my case, it tried to install *Module::Build* (which is a dependency), but then it failed. If this happens to you, simply quit the Perl shell by typing

```
q
```

Then start the Perl shell again:

```
perl -MCPAN -e shell
```

and try to install *Mail::SPF* again:

```
install Mail::SPF
```

This time it should succeed, and you should see that it also installs the modules *Net::DNS::Resolver::Programmable* and *NetAddr::IP* on which *Mail::SPF* depends.

A successful installation of *Mail:SPF* should end like this:

```
Installing /usr/local/bin/spfquery
Writing /usr/local/lib/perl/5.8.8/auto/Mail/SPF/.packlist
  /usr/bin/make install  -- OK
```

Because *NetAddr::IP* has already been installed, we can now leave the Perl shell:

```
q
```

3 Install postfix-policyd-spf-perl

Next we download *postfix-policyd-spf-perl* from **http://www.openspf.org/Software** to the */usr/src/* directory and install it to the */usr/lib/postfix/* directory like this:

```
cd /usr/src

wget http://www.openspf.org/blobs/postfix-policyd-spf-perl-2.001.tar.gz

tar xvfz postfix-policyd-spf-perl-2.001.tar.gz

cd postfix-policyd-spf-perl-2.001

cp postfix-policyd-spf-perl /usr/lib/postfix/policyd-spf-perl
```

Then we edit */etc/postfix/master.cf* and add the following stanza at the end:

```
vi /etc/postfix/master.cf
```

```
[...]
policy  unix -      n       n       -       -       spawn
     user=nobody argv=/usr/bin/perl /usr/lib/postfix/policyd-spf-perl
```

(The leading spaces before *user=nobody* are important so that Postfix knows that this line belongs to the previous one!)

Then open */etc/postfix/main.cf* and search for the *smtpd_recipient_restrictions* directive. You should have *reject_unauth_destination* in

that directive, and right after *reject_unauth_destination* you add *check_policy_service unix:private/policy* like this:

```
vi /etc/postfix/main.cf
```

```
[...]
smtpd_recipient_restrictions = permit_sasl_authenticated,permit_mynetworks,reject_unauth_destination,check_policy_service
[...]
```

or like this:

```
[...]
[...]
smtpd_recipient_restrictions =
        [...]
        reject_unauth_destination
        check_policy_service unix:private/policy
        [...]
[...]
```

that you specify *check_policy_service* *reject_unauth_destination* or else your system can become an open relay!

Then restart Postfix:

```
/etc/init.d/postfix restart
```

That's it already. You should check the *README* file that comes with the *postfix-policyd-spf-perl* package, it contains some important details about how *postfix-policyd-spf-perl* processes emails, e.g. like this part from the *postfix-policyd-spf-perl-2.0001 README*:

This version of the policy server always checks HELO before Mail From (older versions just checked HELO if Mail From was null). It will reject mail that fails either Mail From or HELO SPF checks. It will defer mail if there is a temporary SPF error and the message would othersise be permitted (DEFER_IF_PERMIT). If the HELO check produces a REJECT/DEFER result, Mail From will not be checked.

If the message is not rejected or deferred, the policy server will PREPEND the appropriate SPF Received header. In the case of multi-recipient mail, multiple headers will get appended. If Mail From is anything other than completely empty (i.e.) then the Mail From result will be used for SPF Received (e.g. Mail From None even if HELO is Pass).

The policy server skips SPF checks for connections from the localhost (127.) and instead prepends and logs 'SPF skipped - localhost is always allowed.'

4 Test policyd-spf-perl

We can test *policyd-spf-perl* by running

```
perl /usr/lib/postfix/policyd-spf-perl
```

The cursor will then wait on the *policyd-spf-perl* shell. We can now act as if we tried to send an email from a certain domain and a certain server to another email address. *policyd-spf-perl* will then check if that certain server is allowed to send emails for the sender domain and show us the result.

So let's see what happens if we try to send a mail from *info@h****forge.com* from the server *h****.server*********.net* (IP address *81.169.1**.** *). The *h****forge.com* has an SPF record that allows *81.169.1**.** * to send emails from *h****forge.com*.

So on the *policyd-spf-perl* shell we type:

```
request=smtpd_access_policy

```



```
protocol_state=RCPT

protocol_name=SMTP

helo_name=h****forge.com

queue_id=8045F2AB23

sender=info@h****forge.com

recipient=falko.timme@*******.de

client_address=81.169.1**.**

client_name=h****.server*********.net

[empty line]
```

The output should look like this:

```
action=PREPEND Received-SPF: pass (h****forge.com: 81.169.1**.** is authorized to use 'info@h****forge.com' in 'mfrom'
identity (mechanism 'ip4:81.169.1**.**' matched)) receiver=server1.example.com; identity=mfrom;
envelope-from="info@h****forge.com"; helo=h****forge.com; client-ip=81.169.1**.**
```

which means we passed the test.

Let's run another test, this time we will send from the client `1.2.3.4` (`www.example.com`) which is not allowed to send emails from `h****forge.com`:

```
request=smtpd_access_policy

protocol_state=RCPT
```

```
protocol_name=SMTP

helo_name=h****forge.com

queue_id=8045F2AB23

sender=info@h****forge.com

recipient=falko.timme@*******.de

client_address=1.2.3.4

client_name=www.example.com

[empty line]
```

This is the output, the test failed as expected:

```
action=PREPEND Received-SPF: softfail (h****forge.com: Sender is not authorized by default to use 'info@h****forge.com' in 'mfrom' identity, however domain is not currently prepared for false failures (mechanism '~all' matched)) receiver=server1.example.com; identity=mfrom; envelope-from="info@h****forge.com"; helo=h****forge.com; client-ip=1.2.3.4
```

We can now even try to leave the *sender* field empty, as many spammers do. Still, *policyd-spf-perl* should be able to complete its tests:

```
request=smtpd_access_policy

protocol_state=RCPT

protocol_name=SMTP

helo_name=h****forge.com
```

```
queue_id=8045F2AB23

sender=

recipient=falko.timme@*******.de

client_address=81.169.1**.**

client_name=h****.server*********.net

[empty line]
```

This is the output, we are still allowed to send from *h****forge.com*:

```
action=PREPEND Received-SPF: pass (h****forge.com: 81.169.1**.** is authorized to use 'h****forge.com' in 'helo' identity (mechanism 'ip4:81.169.1**.**' matched)) receiver=server1.example.com; identity=helo; helo=h****forge.com; client-ip=81.169.1**.**
```

Let's try the same test with an invalid client:

```
request=smtpd_access_policy

protocol_state=RCPT

protocol_name=SMTP

helo_name=h****forge.com

queue_id=8045F2AB23
```

```
sender=

recipient=falko.timme@*******.de

client_address=1.2.3.4

client_name=www.example.com

[empty line]
```

As expected, this is the output:

```
action=PREPEND Received-SPF: softfail (h****forge.com: Sender is not authorized by default to use 'h****forge.com' in 'helo'
identity, however domain is not currently prepared for false failures (mechanism '~all' matched))
receiver=server1.example.com; identity=helo; helo=h****forge.com; client-ip=1.2.3.4
```

To leave the `policyd-spf-perl` shell, type

```
[CTRL+C]
```

5 Links

- OpenSPF: **http://www.openspf.org**
- `policyd-spf-perl` Download: **http://www.openspf.org/Software**
- SPF Definition On Wikipedia: **http://en.wikipedia.org/wiki/Sender_Policy_Framework**
- Postfix: **http://www.postfix.org**