Postfix for spamprotection

How to use Postfix's built-in features to reduce the amount of spam and unwanted traffic

Status Quo

Spam is coming from

- open relays
- open proxies
- buggy CGI scripts (e.g. formmail.pl)
- Korea
- China

Status Quo II

Spam is sent

- using faked sender addresses
- containing “paraphrased” key words (V1agra, V!agra, etc.)
- containing “image only” spam
- containing encoded content (e.g. base64 encoded)
- using badly written software

Approaching the problem

- reject mail from open proxies
- reject mail from open relays
- reject mail from servers that run buggy CGI scripts
- reject mail for non-existing users
- insist on RFC-conformance
- reject mail with faked sender addresses
- Use a content scanner

Using Postfix

- use RBLs
- use `local_recipient_maps / relay_recipient_maps`
- configure Postfix to be strict in regard to RFC-conformance
- use RHSBLs
- use sender address verification
- use SpamAssassin

Software Prerequistites

Server

- Postfix
  preferably a recent snapshot version
- caching DNS server
  preferably dnscache

Default Restrictions

Postfix comes with permissive default restrictions:

```
smtpd_recipient_restrictions =
  permit_mynetworks,
  reject_unauth_destination
```

This may not be what you want...

Choosing RBLs

How to choose a RBL?

- examine the listing criteria
  Are they acceptable for you?
- make sure you **know** what is being listed
  hosts, nets, domains, countries
- are the implications clear to you?
  Can you justify using this particular list?

Different RBLs

There are different types of RBLs, they list:

- open relays
- open proxies
- dialup ranges
- countries
- spam sources
- ...

Exceptions

You need exceptions for:

- `postmaster@yourdomain`
- `abuse@yourdomain`

You **must** accept mail to your postmaster and abuse accounts!

You want people to be able to tell you that your restrictions are not working!

Implementation

Extend `smtpd_recipient_restrictions`, by putting the RBLs at the end – they are “expensive”:

```
smtpd_recipient_restrictions =
  permit_mynetworks,
  reject_unauth_destination,
  check_recipient_access
  hash:/etc/postfix/postmaster,
  reject_rbl_client cbl.abuseat.org,
  permit
```

Implementation II

`/etc/postfix/postmaster` contains:

```
postmaster@ OK
abuse@      OK
```

This allows postmaster@anydomain and abuse@anydomain to be reached even if the clients are blacklisted by any subsequent `reject_rbl_client`

More exceptions

What do you do if a customer's server is being blacklisted?

- make an exception
- educate the customer

then

- cry a little

Implementation III

Inserting the exception:

```
smtpd_recipient_restrictions =
  permit_mynetworks,
  reject_unauth_destination,
  check_recipient_access
  hash:/etc/postfix/postmaster,
  check_client_access
  hash:/etc/postfix/cbl_exceptions
  reject_rbl_client cbl.abuseat.org,
  permit
```

Reject mail to unknown users

Advantages

- saves traffic:
  rejection happens at `RCPT TO:`-stage
- saves even more traffic:
  no need to send bounces
- saves yet more traffic:
  What happens if the original sender was fake?

Reject mail to unknown users II

Disadvantages

- list of all valid email addresses needed

Implementation

- If your box is final destination:
  - use `local_recipient_maps`
- If your box is just a relay
  - use `relay_recipient_maps`

Implementation II

Fill `local_recipient_maps / relay_recipient_maps` with the correct data:

```
local_recipient_maps =
  proxy:unix:passwd.byname $alias_maps

relay_recipient_maps =
  hash:/etc/postfix/list_of_users,
  proxy:ldap:/etc/postfix/ask_exchange
```

Require RFC conformance

- sender must be fully qualified
- recipient must be fully qualified
- sender domain must exist
- recipient domain must exist
- require the HELO to be valid

Implementation

The restrictions apply to all clients:

```
smtpd_recipient_restrictions =
   reject_non_fqdn_sender,
   reject_non_fqdn_recipient,
   reject_unknown_sender_domain,
   reject_unknown_recipient_domain,
   permit_mynetworks,
   reject_unauth_destination,
   check_recipient_access hash:/etc/postfix/postmaster,
   check_client_access hash:/etc/postfix/cbl_exceptions,
   reject_rbl_client cbl.abuseat.org,
   permit
```

What is an RHSBL?

- RHSBL stands for:
  “right hand side blocking list”
- The RHS this case is everything to the right of the @ in the sender address.
- An RHSBL is a BL that applies to sender domains, not clients.

Choosing an RHSBL

Same rationale as with the RBLs

To name but a few:

- `dsn.rfc-ignorant.org`
- `postmaster.rfc-ignorant.org`
- `abuse.rfc-ignorant.org`
- `whois.rfc-ignorant.org`

Implementation

Rationale

- RBL/RHSBL lookups are “expensive”
- Perform them as late as possible
- Use a caching DNS server

Implementation II

Again, an exception:

```
smtpd_recipient_restrictions =
   reject_non_fqdn_sender,
   reject_non_fqdn_recipient,
   reject_unknown_sender_domain,
   reject_unknown_recipient_domain,
   permit_mynetworks,
   reject_unauth_destination,
   check_recipient_access hash:/etc/postfix/postmaster,
   check_client_access hash:/etc/postfix/cbl_exceptions,
   reject_rbl_client cbl.abuseat.org,
   check_sender_access hash:/etc/postfix/dsn_exceptions,
   reject_rhsbl_sender dsn.rfc-ignorant.org,
   permit
```

Sender address verification

These are the crown jewels against spam:

- before accepting the mail we can check if it's possible to send mail to the sender
- this is **really** expensive, but results can be cached

How does it work?

- acceptance of the mail is being delayed
- a probe is sent as `address_verify_sender`, unless the sender is cached
- the result is cached in `address_verify_map`
- if the probe doesn't succeed, the acceptance of the mail is rejected temporarily
- meanwhile, the probe is being retried just like “normal” mail

Pitfalls

- some morons send order confirmations as wwwrun@www.bigfirm.de
  - blocking these will make your users more than just unhappy
- some servers accept mail to non-existing users
  - thus rendering any address “valid”

Implementation

- specify a cache
- specify a sender address

```
address_verify_map = btree:/etc/postfix/verify
address_verify_sender = postmaster@example.com
```

- use the restriction `reject_unverified_sender`

Implementation II

insert as a “last resort”:

```
smtpd_recipient_restrictions =
   reject_non_fqdn_sender,
   reject_non_fqdn_recipient,
   reject_unknown_sender_domain,
   reject_unknown_recipient_domain,
   permit_mynetworks,
   reject_unauth_destination,
   check_recipient_access hash:/etc/postfix/postmaster,
   check_client_access hash:/etc/postfix/cbl_exceptions,
   reject_rbl_client cbl.abuseat.org,
   check_sender_access hash:/etc/postfix/dsn_exceptions,
   reject_rhsbl_sender dsn.rfc-ignorant.org,
   reject_unverified_sender,
   permit
```

Selectively applying restrictions

Applying `reject_unverified_sender` for specific sender domains only:

- `smtpd_restriction_classes`

Implementation

- define the `smtpd_restriction_classes`:

```
smtpd_restriction_classes =
  check_if_sender_exists

check_if_sender_exists =
  reject_unverified_sender,
  permit
```

Implementation II

- now use 'em:

```
smtpd_recipient_restrictions =
   reject_non_fqdn_sender,
   reject_non_fqdn_recipient,
   reject_unknown_sender_domain,
   reject_unknown_recipient_domain,
   permit_mynetworks,
   reject_unauth_destination,
   check_recipient_access hash:/etc/postfix/postmaster,
   check_client_access hash:/etc/postfix/cbl_exceptions,
   reject_rbl_client cbl.abuseat.org,
   check_sender_access hash:/etc/postfix/dsn_exceptions,
   reject_rhsbl_sender dsn.rfc-ignorant.org,
   check_sender_access hash:/etc/postfix/suspicious_senders,
   permit
```

Implementation III

- and what's in `suspicious_senders`?
- the usual suspects:

```
hotmail.com check_if_sender_exists
web.de      check_if_sender_exists
gmx.de      check_if_sender_exists
```

Further Reading

- The future of UCE filtering:
  http://dumbo.pobox.com/~mengwong/doc/postfix/#policyd
- Grinch, an open relay checker:
  http://www.zonque.org/projects/grinch
- Tweaking Postfix for high-performance:
  http://kancer.978.org/bulkmailtweaks.htm