Postfix Demystified

A Beginner's Guide to Configuring Postfix

Incoming Mail Restrictions

There was a time in the early days of the internet when spam, phishing, CEO impersonation, email trojans, and other forms of email abuse were not a thing. Mail servers would happily accept mail from any domain and forward it along to any other domain with no worry of being an unwitting accomplice in evildoing. Alas, those blissful days are long gone and, today, a mail server's incoming mail policy is a critical early line of defense against these threats.

In this article, you will learn how to configure Postfix to enforce a simple mail acceptance policy for your domain(s). Your policy is determined by the role(s) your server will play (relay, destination, send-only, etc.), where your users typically connect from, and your tolerance for spam and misbehaving mail clients, among other considerations.

Don't Be an Open Relay!

An *open relay* is a mail server that is configured (misconfigured probably) to relay mail from any sender to any destination, just like the carefree servers of yesteryear. Obviously, open relays are a spammer's dream come true so allowing a server to be an open relay today is the gravest of mail admin sins. Open relays are *always* quickly discovered by spammers, who continuously scan the internet for new servers. Mail services are just as quick to blacklist open servers once they're discovered, but not before damage, possibly permanent damage, has been done to your server's reputation. And once your server's IP address is on a blacklist, it can be difficult to get it removed and you'll find your outgoing mails blocked and dropped by many destination servers until it is. A correctly-configured mail server will help prevent this costly and painful experience.

The SMTP "Conversation"

Before delving into how Postfix applies mail restriction policies, it will be helpful to understand how an incoming host "talks" to your mail server. The set of allowed commands and their responses are called SMTP (Simple Mail Transfer Protocol) and

are defined mostly in RFC 5321. Postfix applies different sets of restrictions at each stage of the SMTP "conversation."

A typical exchange between client (sending host) and server (receiving host) looks like this. In this example, which uses traditional SMTP rather than the more modern ESTMP for brevity, client messages are in normal text and server messages are in **bold**.

```
220 mail.example.com ESMTP Postfix (Debian/GNU)
HELO loser.spamcentral.org
250 mail.example.com
MAIL FROM:<dave.rhodes@spamcentral.org>
250 2.1.0 Ok
RCPT TO:<dlebling@example.com>
250 2.1.5 Ok
RCPT TO:<mblanc@example.com>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
To: dlebling@example.com, mblanc@example.com
From: dave.rhodes@spamcentral.org
Subject: MAKE.MONEY.FAST

My name is Dave Rhodes.  In September 1988 my car was
repossessed and the bill collectors were hounding me like you
wouldn't believe. I was laid off and my unemployment checks
had run out. The only escape I had from the pressure of
failure was my computer and my modem. I longed to turn my
advocation into my vocation. This January 1989 my family and I
went on a ten day cruise to the tropics. I bought a Lincoln
Town Car for CASH in Feburary 1989. . .
.
250 2.0.0 Ok: queued as 2F201211DE
QUIT
221 2.0.0 Bye
```

SMTP commands and responses are sent over TCP port 25 and, including the handshake, the exchange consists of four stages, as seen above:

- HELO/EHLO (handshake): The client and server identify themselves to one another. At this point, postfix begins its investigative work by validating the

client's (alleged) hostname and IP address against DNS records and doing other initial checks.

- MAIL FROM: The client offers the email address of the message sender. Postfix can check the validity and reputation of this *envelope sender* and act accordingly.
- RCPT TO: The client provides the email address(es) for one or more intended mail recipients. Postfix checks that it can handle email for the recipient domains in these *envelope recipients*, sets limits on the number of recipients allowed, and does other checks.
- DATA: The client sends the message headers and body. Postfix checks for a valid message and can perform other checks.

Note from the sample exchange that after each command, the server provides a response. In this example, the response is always "Ok," but if the server detects something amiss or not allowed, it could return an error message and even drop the connection. That's what this article is for: to explain how you can configure Postfix to refuse inappropriate mail while still accepting legitimate messages that you've chosen to allow.

Access Restriction Lists

The table below summarizes each of Postfix's *access restriction lists* – the directives you can include in *main.cf* to enforce your incoming mail policy. Each directive can include a list of rules instructing Postfix to *permit*, *reject*, or *defer* the incoming mail. The directives are evaluated, nominally, in the order listed, with different rules applied at each stage of the SMTP conversation. If a rule in *any* restriction blocks the mail, it will not be accepted.

All directives are optional except that either **smtpd_relay_restrictions** *or* **smtpd_recipient_restrictions** (or both) must exist and must set a policy for handling relay mail (mail not addressed to your domain(s).

Postfix Access Restriction Lists

Restriction List	When it applies*
smtpd_client_restrictions	When the client tries to connect
smtpd_helo_restrictions	After the client says HELO/EHLO
smtpd_sender_restrictions	After the client sends MAIL FROM
smtpd_relay_restrictions	After the client sends RCPT TO

smtpd_recipient_restrictions	After the client sends RCPT TO and after smtpd_relay_restrictions
smtpd_data_restrictions	After the client sends the mail DATA command
smtpd_end_of_data_restrictions	After the client finishes sending the DATA
smtpd_etrn_restrictions	After the client sends ETRN, which can be anywhere but is most common immediately after HELO (uncommon and only applicable to relay servers)

By default, Postfix delays evaluation of rules in the *client*, *helo*, and *sender* restriction lists until after the client sends its first RCPT TO: command, so the server will not normally reject mail until it knows both who it is from and who it is meant for. This is helpful information when reviewing server logs for bounced mail. Because of the delayed rules, many mail admins simply collect all of their rules into smtpd_relay_restrictions and/or smtpd_recipient_restrictions for convenience. This is perfectly fine, though I personally prefer to keep the rules separated by category for clarity in *main.cf*.

The Rules

Each access restriction list has one or more rules separated by commas and/or whitespace. Rules can be on separate lines so long as each new line begins with at least one space or tab. For example, both directives below are valid and equivalent:

```
smtpd_client_restrictions = permit_mynetworks, reject

smtpd_client_restrictions =
     permit_mynetworks
     reject
```

Rules are evaluated in the order they appear (left-to-right or top-to-bottom), so in each of these examples, Postfix will accept mail if it comes from one of its own networks (defined elsewhere in *main.cf*) and will reject all other traffic.

Each rule can instruct the server to *reject*, *permit*, or *defer* an email message. When a message is *rejected*, the server will (usually) return an error code in the 5xx range and refuse to accept the mail, resulting in a bounced message. When the message is *deferred*, the server declines to accept the mail right now, but the message is not bounced and the client may try to send it again later. When a message is *permitted*,

the SMTP conversation continues and the message will be reevaluated at the next restriction list.

It is important to understand that evaluation stops as soon as a rule is matched. Therefore, rule ordering matters. Exceptions to a *reject* rule should always precede the rule they're meant to override. Consider this restriction list, for example (which is real example based on an older message on *Stack Overflow*):

```
smtpd_relay_restrictions =
     permit_mynetworks
     reject_unauth_destination
     permit_sasl_authenticated
```

In this example, the mail admin wondered why his outside SASL-authenticated users couldn't send any outgoing mail. The reason was, of course, because the reject_unauth_destination rule rejected the outside mail with "Relay access denied" before it even had a chance to consider whether the sender was authenticated.

This is not a comprehensive list of all access rules but, rather, includes the restrictions that might be appropriate for most use cases.

Any Restriction List

These rules could appear in any access restriction list, depending on context.

`permit, reject, defer`

These are blanket rules that can be included at the end of a restriction list to make the default policy clear. They are also often used in access tables (see below). By default, each restriction list will *permit* requests that do not otherwise match a rule.

`check_policy_service <servername>`

This rule will invoke an external service to apply a restriction policy not handled natively by Postfix. A common usage is to invoke a service like *policyd-spf* to validate SPF records in with smptd_relay_restrictions. (See my companion article on mail validation for more details.)

`reject_plaintext_session`

When this rule is included, a session is required to use SSL/TLS encryption. This rule should not be used in smtpd_client_restrictions or smtpd_helo_restrictions since those lists may be evaluated before encryption is established.

`reject_unauth_pipelining`

This rule instructs Postfix to reject a request when it is sent out-of-order or without waiting for a response, or when a client tries to use ESMTP pipelining before getting

a proper EHLO response. It can foil some spammers who try to bulk-send emails without proper give-and-take.

warn_if_reject

This modifier should only be used in a testing environment. It overrides the following *reject* rule and causes a warning to be logged instead.

smtpd_client_restrictions

These rules are evaluated against the client's actual IP address but, by default, they are not applied until after the first RCPT TO: command is issued. It is acceptable to put these rules in smtpd_relay_restrictions or smtpd_recipient_restrictions if you prefer. By default, Postfix will accept connections from all clients. Most basic Postfix configurations do not need a smtpd_client_restrictions directive.

check_client_access <*type*>:<*client_access_table*>

This rule asks Postfix to consult an access table and make a decision about how to proceed based on the client's IP address. This can let you set IP-specific policies. See the section about access tables before for more details.

reject_unknown_client_hostname

Postfix performs a reverse-lookup for the client's IP address and then (if a valid domain name is returned) a forward-lookup on the domain name. This rule will reject the connection under any of the following conditions:

- The reverse-lookup did not return a valid domain name.
- The forward-lookup did not return a valid IP address.
- The resultant IP address does not match the client's IP address.

smtpd_helo_restrictions

These rules are evaluated against the provided HELO/EHLO hostname but, by default, they are not applied until after the first RCPT TO: command is sent. It is acceptable to put them in smtpd_relay_restrictions or smtpd_recipient_restrictions if you prefer. By default, Postfix will tentatively accept mail from all HELO/EHLO hostnames.

Note that, by default, Postfix does not require a HELO/EHLO command, so a misbehaving client could bypass these checks by simply not admitting who they are. To require a HELO/EHLO command, add this directive to *main.cf*:

```
smtpd_helo_required = yes
```

`check_helo_access <type>:<helo_access_table>`

This rule asks Postfix to consult an access table and make a decision about how to proceed based on the client's provided hostname or IP address. See the section about access tables before for more details.

`reject_invalid_helo_hostname`

Rejects the HELO/EHLO command if the hostname is not a correctly-formed valid domain name or IP address.

`reject_unknown_helo_hostname`

Rejects the HELO/EHLO command if a forward-lookup on the provided hostname does not return an IP address.

smtpd_sender_restrictions

These rules are evaluated against the *envelope sender* address, which is provided by the client's MAIL FROM: command. By default, they are not applied until after the first RCPT TO: command is sent. It is acceptable to put these rules in smtpd_relay_restrictions or smtpd_recipient_restrictions if you prefer. By default, Postfix will tentatively accept mail from all sender addresses. Most basic Postfix configurations do not need an smtpd_sender_restriction directive.

`check_sender_access <type>:<sender_access_table>`

This rule asks Postfix to consult an access table and make a decision about how to proceed based on the envelope sender address. See the section about access tables before for more details.

Additional rules for preventing your own users from spoofing their envelope sender addresses can be applied. See my companion article about managing your mail users for more information about these rules.

smtpd_relay_restrictions and smtpd_recipient_restrictions

These rules are applied against each *envelope recipient* address, which is provided by the client's MAIL RCPT: command(s), with smtpd_relay_restrictions be evaluated first and smtpd_recipient_restrictions after by default. Note that the rules in both restriction lists are evaluated for all incoming mail, regardless of its destination.

By default, Postfix will allow:

- all mail from your networks (defined by *mynetworks* in *main.cf*) and all mail from SASL-authenticated users, as well as
- all mail addressed to domains you handle (defined by *inet_interfaces*, *proxy_interfaces*, *mydestination*, *virtual_alias_domains*, and

virtual_mailbox_domains in *main.cf*) or relay to (defined by *relay_domains* in *main.cf*).

`check_policy_service <servername>`

This rule will invoke an external service to apply a restriction policy not handled natively by Postfix. Depending on the policy, it might be included in any restriction list, but is commonly-used here (or sometimes in stmpd_sender_restrictions) with *policyd-spf* to validate the incoming host against the sender domain's SPF record and apply that domain's SPF policy. (See my article on mail validation for more details.) Putting it in recipient_sender_restrictions ensures that the intended recipient is known and logged.

`check_recipient_access <type>:<table>`

This rule asks Postfix to consult an access table and make a decision about how to proceed based on the envelope recipient address. See the section about access tables before for more details.

`check_sasl_access <type>:<table>`

This rule is a little more obscure than some of the others I've included. It asks Postfix to consult an access table and make a decision based on the SASL login name of a user. It can come in handy if you want to exclude certain users from a *permit_sasl_authenticated* rule that follows it. See the section about access tables below for more details.

`permit_mynetworks`

Explicitly allow connections and mail from any client IP address that you have specified in the *mynetworks*= list elsewhere in *main.cf*. By default, *mynetworks*= includes only localhost. This should precede a *reject* or *check* rule when you want to make an exception for users on your own network(s).

`permit_sasl_authenticated`

Explicitly allow connections and mail from any client that has authenticated with a username and password using SASL. This should precede a *reject* or *check* rule when you want to make an exception for your own authenticated users.

`reject_rbl_client <rbl_domain>`

This rule instructs Postfix to query an RBL (remote block list/realtime blackhole list) and reject the mail if the RBL reports that the client IP address is known to send malicious email. Although this rule *can* be included in the *client*, *helo*, or *sender* restriction lists, it is good practice to include it here instead. Doing so ensures that the RBL query, which takes extra time and resources, is only performed for connections that aren't already rejected by earlier less resource-intensive checks.

Spamhaus and SpamCop are two commonly-used RBL maintainers and can be included with

```
reject_rbl_client zen.spamhaus.org
reject_rbl_client bl.spamcop.net
```

Be aware, however, that blackhole lists can contain false positives, which could cause your server to reject legitimate emails. Mail servers hosted on cloud VPS's (especially those with dynamic IP addresses and even more especially those on Digital Ocean or AWS) or (heaven forbid) home internet connections are particularly prone to this. For this reason, you should include permit_sasl_authenticated before this rule, even if you think you don't need to. Also, be aware that some RBL maintainers are more trustworthy than others. The blackhole lists provided by UCECONNECT, for example, have a reputation for being particularly overzealous.

This rule is most often included under smptd_recipient_restrictions so that it occurs as late as possible in processing, but before the DATA command.

`reject_unauth_destination`

This is the single most important rule for ensuring that your server is not an open relay (see below). It instructs Postfix to reject any mail whose *envelope to* address does not match an authorized domain. Authorized domains include mail the server actually handles (defined by *inet_interfaces*, *proxy_interfaces*, *mydestination*, *virtual_alias_domains*, and *virtual_mailbox_domains* in *main.cf*) or is authorized to relay to (defined by *relay_domains* in *main.cf*).

However, you probably want your own authenticated users to be able to send mail to outside domains, so be sure to include permit_mynetworks (for local users who do not need to authenticate over STMP) and permit_sasl_authenticated (for users who do authenticate over SMTP) before reject_unauth_destination.

This rule is most often included under smtpd_relay_restrictions.

`reject_unauth_pipelining`

This rule instructs Postfix to reject a request when it is sent out-of-order or too quickly, or when a client tries to use ESMTP pipelining before getting a proper EHLO response. It foils some spammers who try to bulk-send emails without proper give-and-take. While this rule can be included in any restriction list, it is usually first included in smtpd_relay_restrictions or smtpd_recipient_restrictions since that will include all earlier attempts at unauthorized pipelining.

smtpd_data_restrictions

reject_unauth_pipelining

This rule *still* instructs Postfix to reject a request when it is sent out-of-order or when a client tries to use ESMTP pipelining before getting a proper EHLO response. In smtpd_data_restrictions, it can help prevent SMTP smuggling.

smtpd_end_of_data_restrictions

This restriction list is not necessary in most normal Postfix configurations, but some systems might want to include a check_policy_service rule here to enforce a policy limiting message length or imposing quotas, for example.

smtpd_etrn_restrictions

This restriction list is not necessary in most normal Postfix configurations as ETRN is not commonly-used in today's always-online age.

Common Examples

A common "generic" configuration is included here. In this sample, Postfix will reject mail from clients that don't report their hostname, or report an unintelligible one, as well as from hosts who send multiple commands in a single data packet. It will allow your local and authenticated users to send mail to one another and to outside destinations. It will refuse to relay any messages from others.

```
smtpd_helo_required = yes

smtpd_helo_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_invalid_helo_hostname,
    reject_unknown_helo_hostname

smtpd_relay_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_pipelining,
    reject_unauth_destination
```

Since Postfix doesn't apply *client*, *helo*, or *sender* rules until the RCPT TO: command, you can simplify the above by combining them together. This ruleset is equivalent to the one above:

```
smtpd_helo_required = yes

smtpd_relay_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_invalid_helo_hostname,
    reject_unknown_helo_hostname,
    reject_unauth_pipelining,
    reject_unauth_destination
```

It is also common to include rules to block known spammers. To reject mail from clients on the Spamhaus and SpamCop realtime blackhole lists, you can add this directive to either of the above.

```
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_rbl_client zen.spamhaus.org,
    reject_rbl_client bl.spamcop.net
```

If your mail server is used *only* to send outgoing mail and will never receive mail from the outside, you can accomplish that with this example.

```
smtpd_client_restrictions =
    permit_mynetworks,
    reject
```

Access Tables

Sometimes, you might want to create conditional rules that only apply to certain clients, certain hosts, or certain addresses. You can do that with access tables. In the simplest case, an access table is a text file that includes one rule per line. Each rule includes a pattern to search against, followed by whitespace, followed by the action to take when that pattern is matched:

```
<pattern>      <action>
```

Pattern can be an IP address, a domain name, or an email address, depending on the specific access map. Wildcards are allowed by omitting parts from the pattern. For example, “@example.com” will match any user with the domain *example.com* and “111.222.333” will match any IP address on the subnet 111.222.333.0/24.

Access tables are processed top-to-bottom and, generally, all matching actions are applied (but this can vary depending on the action). When a match is found, *action* is carried out. The most common *actions* include:

`REJECT`	Reject the mail
`BCC <address>`	Send a copy of the mail to *address*
`REDIRECT <address>`	Send the mail to *address* instead of its original recipient
`DISCARD`	Accept the mail, then silently make it disappear
`PREPEND <name>:<value>`	Prepend a header of the specific *name* with the specified *value* at the top of the message.
`DUNNO`	Return a "no match" response.
`<restriction>`	Return the action of another rule, such as *permit_mynetworks* or *defer*

For a complete list of valid actions and their uses, see the Postfix Access Man Page.

Note that the action "OK," while valid for Postfix access tables in general, is not appropriate for restriction list access tables. Instead, a value of "DUNNO" should be returned which, in this context, simply means "do not reject the mail based on this rule alone."

Once an access table's text file has been written, you will need to digest it into a hashed file with the *postmap* command:

```
postmap <filename>
```

This will create *filename.db*, which can be included as an argument of type "hash" to a *check_xxx_access* rule in a restriction list. For example, to include the hashed file "sender_access.db" in a check_sender_access rule, you would use:

```
Smtpd_sender_restrictions =
     postmap check_sender_access hash:sender_access
```

Note that you should not include the .db extension in the directive.

Examples

To silently discard emails sent a your domain's "noreply" address:

```
# main.cf
smtpd_recipient_restrictions =
  check_recipient_access hash:/etc/postfix/recipient_access

# recipient_access
noreply@example.com DISCARD
```

To prohibit outsiders from sending emails that purport to be from your domain:

```
# main.cf
smtpd_sender_restrictions =
  check_sender_access hash:/etc/postfix/sender_access

# sender_access
@example.com   permit_mynetworks
               permit_sasl_authenticated
               REJECT
```

To block access from a subnet, but allow one particular whitelisted IP through:

```
# main.cf
smtpd_client_restrictions =
    check_client_access hash:/etc/postfix/client_access

# client_access
111.222.333.444     DUNNO
111.222.333         REJECT
```

Rate-Limiting Abusive Clients

In addition to defining restriction lists, you can make some adjustments in *main.cf* that can slow spammers down. If you choose to adjust these directives, set them to high values and take care not to restrict legitimate incoming mail. Most mail admins will leave these values at their defaults.

`smtpd_client_message_rate_limit = <limit>`

This option will restrict the number of messages that a client can send per minute. The default is unlimited.

`smtpd_client_recipient_rate_limit = <limit>`

This option will restrict the number of recipient addresses that a client can announce per minute. The default is unlimited.

`smtpd_recipient_limit = <limit>`

This option will restrict clients to *limit* number of recipients per message. The default is 1000 recipients.

`smtpd_soft_error_limit = <limit>`

The number of errors that a client is allowed to make without successfully sending mail before Postfix begins throttling the client's requests. The default is 10 errors.

`smtpd_hard_error_limit = <limit>`

The number of errors that a client is allowed to make without successfully sending mail before Postfix drops the connection. The default is 20 errors.

`smtpd_error_sleep_time = <time>`

The amount of time (seconds by default) that Postfix will pause after each error when the number of errors is greater than *smtpd_soft_error_limit* and less than *smtpd_hard_error_limit*.

`smtpd_junk_command_limit = <limit>`

The number of non-mail-related commands the client can send before Postfix begins counting them as errors. The default is 100 junk commands.

Again, Don't Be an Open Relay!

Fortunately, there are free services on the internet that can check your mail server's security and alert you to possible problems. One well-known service is mxtoolbox.com. Their SMTP diagnostics test will try to relay a test message through your server and warn you if they succeeded. You should run such a test any time you make significant changes to your mail server configuration.

This article has given an overview of the most common configuration options that will apply to most new mail servers. If your needs are more complex or you want to learn more, you should refer to the official Postfix documentation:

- Postfix SMTP Relay and Access Control
- All Postfix Configuration Parameters
- Access Tables Man Page
- User-Defined Restriction Classes

Everything on the internet is ephemeral. Please let me know if you discover any broken links.