

TrustPath

[SpamAssassin](#), by default, will automatically attempt to figure out which Received: headers were inserted by mailservers in your network, and which were not. This auto detection works pretty well for most networks, but when Network Address Translation (NAT) is involved there are two possible configurations that [SpamAssassin](#) cannot automatically detect the difference between. One case is where your MX has a publicly routed IP, and the other where it does not. Looking at Received: headers alone, SA cannot distinguish between the two, so it assumes your MX has a public IP. Also, if you have a large or complex network topology, it's strongly recommended you set this (most ISPs, for instance).

The common symptoms of a broken Trust path include:

- [ALL_TRUSTED](#) matching spam email from the outside or other untrusted mail.
- Dialup/Dynamic IP RBLs misfiring for properly relayed mail.
- Dialup/Dynamic IP RBLs not catching direct-delivered mail.
- `whitelist_from_rcvd` fails to match.
- SPF tests misfiring (failing when they should pass and vice versa).
- False positives on non-spam mail coming from "dynamic" or "dialup" addresses in your own network.
- [AutoWhitelist](#) mismatches on forged mail due to confusion about the source IP.

Your trust path can be tested by adding the following to your SpamAssassin config:

```
add_header all RelaysUntrusted _RELAYSUNTRUSTED_
```

This will add headers similar to "X-Spam-RelaysUntrusted: [ip=140.211.11.3 rdns=hermes.apache.org...". The first IP address is the IP which will be used for network tests like RBLs and SPF.

If you see these warning signs frequently, you probably need to manually configure `trusted_networks`. See the [Mail::Spamassassin::Conf](#) manpage for details. Generally you want `trusted_networks` set to contain all the mailservers you control that add Received: headers, and nothing else. For proper operation of DUL and SPF tests on authenticated mail submission from dynamic/"dialup" hosts, see [DynablockIssues](#).

Here's an example `trusted_networks` line that could be added to `/etc/mail/spamassassin/local.cf` to specify trust:

```
trusted_networks 123.12.34.56 123.12.35/24 127.0.0.1
```

That line will specify that the host at 123.12.34.56, the local host 127.0.0.1, and all hosts in the 123.12.35.0 - 123.12.35.255 address range, are to be trusted. (In [SpamAssassin](#) 3.2.x, it will no longer be necessary to specify 127.0.0.1; it'll automatically be trusted implicitly.)

Why doesn't [SpamAssassin](#) default to not trusting any hosts?

Well, trusting too few is in many ways just as bad as trusting too many. Many [SpamAssassin](#) rules try to perform checks against the untrusted host that delivered mail to the first trusted server. If there's too few or too many hosts that SA trusts, these tests will be examining the wrong host. Both situations contribute greatly to false negative problems, and to a lesser extent false positive problems.

Why doesn't the auto detection just work for all networks?

Unfortunately there's limits to what one can automatically discover about a network from just email headers.

It's pretty obvious that any [RFC 1918](#) (which obsoleted [RFC 1597](#)) private IP's in the most recent Received: header are part of the local network. From there, tracking backwards in terms of time, each additional private IP can be safely assumed to be a part of the local network until you hit the first non-private IP.

The problem is how can you tell if that first non-private IP is part of the local network? You can't.

You could be dealing with a network in which the gateway MX is part of a DMZ which is not address translated, in which case that non-private IP is local. On the other hand, you could be dealing with a network where the gateway MX is address translated, in which case that non-private IP is outside the local network.

Thus, [SpamAssassin](#) has to make an assumption. That assumption works for some NAT networks, and not for others. You can change the assumption, but at that point you're just changing which of the two ambiguous cases will need manual configuration.

If you really want to be on the safe side, just declare your `trusted_networks` manually, and you'll avoid the auto detection situation entirely.

How can I optimize the `trusted_networks` setting?

If you want to configure [SpamAssassin](#) with more information, you can:

- set 'internal_networks' to include the hosts that act as MX for your domains, or that may deliver mail internally in your organisation.
- set 'trusted_networks' to include the same hosts and networks as 'internal_networks', with the addition of some hosts that are external to your organisation which you trust to not be under the control of spammers. For example, very high-volume mail relays at other ISPs, or mailing list servers. Note that it doesn't matter if the server *relays* spam to you from other hosts; that still means you trust the server *not to originate spam*, which is what 'trusted_networks' specifies.

By giving [SpamAssassin](#) more info about your network setup, it can perform some tests better, increase accuracy, and reduce load.

Can you explain more about what [SpamAssassin](#) does with this info?

Take a look at [TrustedRelays](#).

Note that the trust path information is used by both network and non-net rules, so even if you're not running with network rules enabled (the "-L" switch), it's worth configuring this.

Note that each of `trusted_networks` and `internal_networks` default to taking the value of the other if unset. See the `Mail::SpamAssassin::Conf` man page for details. Therefore once either is set then either must be fully declared. For example if several IPs are placed in `trusted_networks` that value list will be defaulted into `internal_networks` if `internal_networks` is not set. But if `internal_networks` is set to even a single IP value then this will not occur. In that case processing which depends upon a correct `internal_networks` configuration such as `RCVD_IN_SORBS_DUL` will not be triggered. If you set both `trusted_networks` and `internal_networks` then both must be fully specified. A simpler configuration is to set only the `trusted_networks` parameter and allow the `internal_networks` parameter to default to that value. An easy trap to fall into is to only partially specify one or the other.

Why should `trusted_networks` and `internal_networks` ever be different?

A mail relay that you want to trust in `trusted_networks` may itself trust its own internal dynamic IP networks. You may trust them not to be a spam source but putting them into your `internal_networks` list would create a false positive because then those dynamic IPs would be searched for in the DUL lists. This is an example where the two lists need to be different.