Enhanced Spam Defence

An approach to making SMTP connect time blocking a reliable method for e-mail filtering

By John Jensen, Topsec Cloud Solutions Ltd.

As the spam problem keeps growing and the associated threats become ever more dangerous to the stability and performance of a company's e-mail systems, the steps taken to combat these threats must also constantly be developed.

New and enhanced virus scanning systems are developed, more advanced content scanning and text analysis applications are brought to market, image analysis software is implemented, all with varying degrees of success.

All of these systems share a common approach: trying to analyse the contents of the e-mail in either text, binary or image form.

An obvious problem with this approach is that this type of content analysis is largely a subjective evaluation of what should be considered spam – with a high risk of blocking an e-mail which shouldn't be blocked, creating a false positive.

Although some systems of this type deploy a form of e-mail signature recognition, similar to virus scanning systems, and thus rely on recognising e-mails already seen and identified as spam, they struggle to capture e-mails containing random text or images with variations in pixel composition.

It is exactly these characteristics that are increasingly common in current spam e-mails.

Therefore a different approach is needed.

What we need to do is to apply filtering to something other than content, without introducing the additional risk of false positives.

SMTP connect time blocking has been considered by some to be an unreliable or risky method for e-mail filtering.

This has mainly been because the existing solutions attempting this approach have been trying to deliver an all-in-one application that attempted both SMTP connect time blocking as well as content analysis in one step.

In the following I will describe how it is possible to configure a solution that makes SMTP connect time blocking a very reliable method for e-mail filtering.

The basic principle applied is that we do not try to confirm something to be true – instead we perform checks which may confirm that information given to us is false. When this happens we can safely drop the connection without risking a false positive.

It is important that the SMTP connect time checks are performed as the very first checks applied to all e-mail. For this reason, this system should be used on an e-mail gateway which is directly connected to the Internet. This is normally the machine which is pointed to by a domain's MX records.

To implement this principle in our gateway system it is important that the checks are carried out in a particular order. Not all checks are appropriate in all cases and some special consideration may be given to certain checks before they are applied to a particular e-mail.

We want to perform the simple, straightforward checks first and then gradually apply increasing complexity. This way we save time and processing power on our gateway system and at the same time we achieve a higher level of accuracy.

Every e-mail can be broken down into two parts: 1) the SMTP header and 2) the contents or DATA portion.

For the connect time checks we only need to use the information in the SMTP header part of the e-mail and we do not need to receive the DATA portion at all.

This has obvious advantages; firstly that e-mails blocked by SMTP connect time checks are never actually received and we thereby achieve a significant bandwidth saving, and secondly that we do not risk that any contents or attachments pollute our content analysis filters.

The SMTP header is further broken down into at least three parts: 1) HELO/EHLO information from the sending server, 2) MAIL FROM containing the from e-mail address and 3) RCPT TO containing the recipient e-mail address.

In addition to these three pieces of information a number of other headers may exist containing information about servers that the e-mail has been routed through, stamps from scanning systems, etc. – all of which may be genuine or forged.

The following simplified example is using Postfix to demonstrate how a number of checks can be applied at connect time to the three SMTP headers – HELO/EHLO, MAIL FROM and RCPT TO – to allow us to decide whether to reject or accept the connection.

The relevant checks are implemented on the `smtpd_recipient_restrictions` section of Postfix's main configuration file.

```
smtpd_recipient_restrictions =
1 reject_unauth_destination,
2 check_helo_access hash:/etc/postfix/helo_access,
3 reject_invalid_hostname,
4 reject_non_fqdn_sender,
5 reject_unknown_sender_domain,
6 check_sender_access hash:/etc/postfix/sender_access,
7 reject_non_fqdn_recipient,
8 reject_unknown_recipient_domain,
9 reject_unauth_pipelining,
10 reject_rbl_client sbl.spamhaus.org,
11 check_sender_access hash:/etc/postfix/frequently_forged,
12 check_client_access regexp:/etc/postfix/greylisted
```

Note: This is not the full configuration file, additional settings are required. Configuration of Postfix is beyond the scope of this article.

Line 1: First make sure that the system is not an open relay that can be used to send spam.

Line 2: Then we can apply checks to the HELO/EHLO information that the sending server has given to us.

This check is implemented as a lookup table; in this simple example it is a hash table but it could also easily be a table in an SQL database.

At this stage in the SMTP conversation the sending server has given us its name and we can now verify whether this name is deliberately wrong. A large proportion of spam engines use the name or IP address of the server it is connecting to and this can easily be checked.

A server sending mail to us should never use our name as a HELO/EHLO name and anyone that does, will not be allowed to connect.

The contents of this table should be something like the following:

```
localhost               550 You are not localhost
127.0.0.1               550 You are not 127.0.0.1
friend                  550 friend name not valid
62.77.162.9             550 You are not 62.77.162.9
topsec.com              550 Sending host claims to be topsec.com
```

550 is the reject code indicating a permanent rejection. Codes starting with 4xx are temporary reject codes and the sending server may try to deliver the message at a later time. Letting the sending server retry the delivery at a later time is also a technique we can use with great effectiveness later in the process – an example below is the greylisting configuration.

Line 3: If the connection request makes it through the first check – i.e. it doesn't claim to be us – we check if the hostname given is in a valid format. If not, then the connection is dropped.

Next we can get to work on the information given about the e-mail's "from" address.

Line 4: First we reject any e-mail where the sender address is not in the form of address@domain.com.

Line 5: Then we reject e-mail where the sender address is in a domain that does not exist. This is verified with a DNS lookup – if there is no registration for the domain, then the connection is rejected.

Line 6: In the same way that servers can use our name, sender addresses can also pretend to be from our domain in an attempt to establish connections. To catch these attempts we need to check the sender information in the same way as we did with the server information.

We use a lookup table that looks like this:

```
topsec.com              550 Mail claims to be from topsec.com
```

All of our own domains should be in this table.

Line 7 – 9: We want to reject e-mails to recipient addresses that are not in the correct format or unknown, and we don't want to allow too many simultaneous delivery attempts through our server.

Line 10: For connections that survive this far, we may now wish to check the sending server's IP address against one or more blacklists. Be very careful when selecting a blacklist as many are not well maintained and therefore inaccurate, causing false positives.

In this example we are using a blacklist from spamhaus.org which is updated constantly and very accurate.

If you are running your own blacklist, it is very important to keep it updated and to regularly delete entries older than a few hours or days.

Line 11: Sender address verification is a technique that must be used sparingly and only when planning for its limitations.

The sender verification process opens a connection back to the MX server for the sender's domain and attempts to deliver an e-mail to the sending e-mail address.

Many domains route their e-mail through a third party and in many cases the third party may not have a list of valid e-mail addresses and will simply accept e-mail to any address in the domain. This means that even if the server accepts the address verification attempt the address may still not exist.

For performance reasons it is not a good idea to try to verify every single e-mail that we receive.

Instead we want to limit address verification to domains that we know are frequently used as forged addresses in spam e-mails.

We can repeat the `check_sender_access` statement but with a different table; in this case we use the `frequently_forged` table with the following contents:

```
aol.com     reject_unverified_sender
hotmail.com reject_unverified_sender
```

Just add the domains to which the address verification process should be applied.

We can then store the result of each address verification attempt so that it can be reused in the future. This saves time as we then don't have to perform the full verification process for the same addresses.

It is important to let the verification results have only a limited lifespan, as e-mail addresses are constantly created and deleted.

Line 12: As mentioned earlier, a 4xx reject code causes a connection to be temporarily rejected and the sending server will then queue the e-mail and retry delivery later, typically after 5 to 30 minutes.

This is standard behaviour of all e-mail servers, but most spam is delivered not through e-mail servers but from compromised desktop and home user PCs that have been infected with some form of worm or other malicious programme.

The main concern for spammers is to be able to deliver large amounts of spam very quickly, and that frequently means that their spam sending software does not handle temporary reject codes correctly.

Simply by temporarily rejecting a connection from a spamming PC, the spam e-mail can be blocked as the delivery will not be retried.

Legitimate e-mail will not be lost; it will simply be delayed for a few minutes until the sending server retries delivery.

As this technique does introduce delays to e-mails, it should be applied very selectively and only to senders that appear not to be genuine e-mail servers.

A recommended approach is to use the name returned to us by performing a reverse DNS lookup on the IP address which has established the connection to us. This gives us the client name, which is independent of the HELO/EHLO information given to us earlier.

In this case the `check_client_access` statement is used with a table that contains a set of regular expressions.

The table in this example looks like this:

```
/unknown/       greylisting
/br(e|oa)dband|cable|dial?(in|up)?|home|in-addr|ppp|ptr/      greylisting
```

These regular expressions are then executed against the client name returned as a PTR record from the reverse DNS lookup.

What we are looking for are IP addresses that do not resolve (PTR record returns ‘unknown’) or IP addresses that appear to be on dial-up, DSL or home user networks, as these are the types of PCs most frequently compromised by spamming engines.

When used in this way, greylisting is a very effective technique that blocks a large amount of spam without creating delays for genuine e-mails.

In summary:
We have seen how a selective and targeted application of SMTP connect time blocking techniques can provide a very effective and reliable method for e-mail filtering on an e-mail gateway system.

The techniques described here – together with some others beyond the scope of this article – are in production use and have proven to catch in excess of 97% of spam. Used in combination with traditional content analysis anti-spam applications running a second stage of scanning on separate servers behind the gateway system, this approach provides a very effective and accurate spam blocking mechanism which is able to keep up with spammers and the constant changes in their tactics.

About the author:
John Jensen is the CTO of Topsec – an Irish owned company, which is part of the Top Security group, operating in Ireland, South Africa and the UK and offering Topsec Email Security, an e-mail filtering service, as well as Internet security services.