Protecting A Dialin Infrastructure From Spammers

Back in 2002, the guys at init7 have developped a concept to protect anonymous dial-in from being abused by spammers.

The original concept as outlined here consists basically of three combined efforts:

#1 Redirect SMTP Connections to a SMTP proxy on the core router

#2 Enforce rate limits on the SMTP proxy

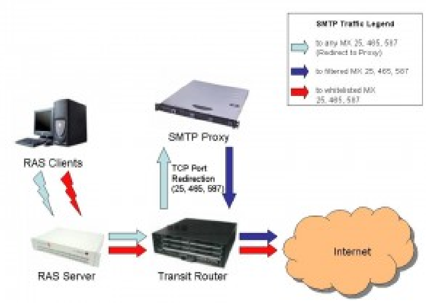
#3 Temporary reject source IP which have exceeded their limits

Inspired by the basic concept I started to implement it at our site.

Soon enough I should come along some caveats causing me to change things a bit while still retaining most of the original concept.

The Concept Reloaded

Please click on the thumbnail image to get a larger view of the concept drawing.


SMTP Traffic Legend
to any MX 25, 465, 587 (Redirect to Proxy)
to filtered MX 25, 465, 587
to whitelisted MX 25, 465, 587
SMTP Proxy
RAS Clients
TCP Port Redirection (25, 465, 587)
RAS Server
Transit Router
Internet

Compared to the original concept I had some additions in mind.

- Support not only TCP port 25 (SMTP) for proxy redirection but also include ports 465 (SMTP/TLS) and 587 (SMTP/Submission)
- Use a separate SMTP server as proxy gateway
- Allow unrestricted (non proxied) access to internal SMTP hosts for authorized users
- Use native rate-limiting features of Postfix MTA
- Use a native DNS RBL for temporary blacklisting instead of MTA-specific hash tables

Assumptions made in my examples:

- RAS IP Range is 172.16.1.0/25
- First allowed MX Host is 192.168.1.27 (outbound)
- Second allowed MX Host is 192.168.1.28 (listserver)
- SMTP Proxy Host is 192.168.1.29
- SMTP Proxy host runs FreeBSD
- Disallowed MX Hosts is 0.0.0.0 (smtp)
- Transit Router is a Cisco

Transit Router Configuration

First of all you need to configure the redirection scheme on your transit router. I implemented it using access lists and a combined route map.

Given my case I had two SMTP servers to allow connections to in any case (security policy implemented via SMTP authentication scheme) and all others to be filtered. This is how it is implemented:

```
ip access-list extended dialin_to_listserver
permit tcp 172.16.1.0 0.0.0.127 host 192.168.1.28 eq smtp 465 587
remark match smtp connections from dialin range to listserver

ip access-list extended dialin_to_outbound
permit tcp 172.16.1.0 0.0.0.127 host 192.168.1.27 eq smtp 465 587
remark match smtp connections from dialin range to outbound server

ip access-list extended dialin_to_smtp
permit tcp 172.16.1.0 0.0.0.127 any eq smtp 465 587
remark match smtp connections from dialin range to any host
```

Now these access lists are combined into a route map. The route-map first matches against the allowed MX hosts (preference 10 and 15) and then against all unspecified hosts (preference 20). The last rule causes unspecified connections to be redirected to the smtp proxy host by overriding it's next-hop.

```
route-map SMTP-proxy permit 10
description do not force ip redirect for smtp connections to outbound server
match ip address dialin_to_outbound
!
route-map SMTP-proxy permit 15
description do not force ip redirect for smtp connections to listserver
match ip address dialin_to_listserver
!
route-map SMTP-proxy permit 20
description force ip redirect for unspecified smtp connections
match ip address dialin_to_smtp
set ip next-hop 192.168.1.29
!
```

Finally the route map is attachend to the outbound interface:

```
interface GigabitEthernet0/1
ip policy route-map SMTP-proxy
```

SMTP Host: IPFW Configuration

The SMTP proxy host requires that you create a special accept rule for the forwarded packets to be handled. A FreeBSD host requires ipfw with IP_FORWARD to be enabled in the kernel for this work.

```
add 11 fwd 192.168.1.29,25 tcp from 172.16.1.0/25 to any 25
add 12 fwd 192.168.1.29,465 tcp from 172.16.1.0/25 to any 465
add 13 fwd 192.168.1.29,587 tcp from 172.16.1.0/25 to any 587
```

Consider that a "check-state ip from any to any" rule should not be invoked before any 'fwd' rules, otherwise the packets will be rejected as connection tracking is not supported on forwarded packets.

SMTP Host: Postfix Configuration

All newer versions of Postfix support rate limiting through anvil(8). Anvil allows easiliy to keep track of connection and recipient numbers and reject clients if they exceed their limits.

Anvil can be enabled by adding some statements to main.cf:

```
smtpd_client_connection_count_limit=50
smtpd_client_connection_rate_limit=50
smtpd_client_event_limit_exceptions=127.0.0.1
smtpd_client_message_rate_limit=50
smtpd_client_recipient_rate_limit=50
anvil_rate_time_unit=1800s
```

More complex configurations, eg. for different limits per source IP or subnet can be created using policy based routing in Postfix, though this is beyond the scope of this tutorial.

SMTP Host: RBL support

Given the case a malicious client exceeds the rate limits set in the previous step, Postfix's anvil will automatically reject the client until the timeout interval is reached (1800s in the example above).

While this should stop immediate abuse, an attacker could try to adjust his delivery limits to stay below our treshold.
This can be avoided if a client's IP address was to be blacklisted temporarily upon exceeding the limits. For interopability reasons I have written the 'dsb' plugin for my logtail daemon (both to be released shortly), which basically monitors the maillog and adds any client IP exceeding the rate limits to an internal DNS blacklist.

I choose not to use classic MTA-specific hash tables as DNS is more portable and can easily be used in a distributed environment.

To enable realtime blacklisting support, add a statement to your smtpd client restrictions in Postix's main.cf:

```
smtpd_client_restrictions = permit_mynetworks,reject_rbl_client myrbl.mydomain.tld
```

Additionally you'll need to setup a blacklist zone (myrbl.mydomain.tld in the exmaple) on your favorite DNS server (ISC BIND strongly recommended, server must support dynamic zone updates for full compatibility with logtail + dsb).

Then install logtail + dsb plugin on your SMTP proxy server and enable it for automatic startup. For logtail + dsb visit http://phaq.phunsites.net/dsb (software to be released soon).

Radius Server: RBL unlisting support

If you chose to use logtail + dsb for automatic IP blacklisting support, this addition may come in handy. The logtail + dsb package includes a handy little tool which is to be added to your Radius server to allow for automatic unlisting an IP address upon client disconnect.

This will workaround an IP from your dynamic pool staying blocked even if the malicous client disconnected ages ago.