Minimising Spam

How to slay spam and reclaim your resources

Andrew Richards, August 2009 for UKUUG conference.

Outline

- How does spam burden my system resources?
- What can I do about it – and when?
- SMTP: Envelope & message
- Spammers, viruses & SMTP compliance
- Lightweight measures to identify spam by behaviour or source
- Other 'early' techniques
- Seeing these measures in action
- Testing your own system
- Real-world systems
- Conclusions, questions

How does spam burden my system resources?

Typically > 90% of emails received by mail servers can be spam, so for every valid email you receive, you have to handle at least another 9.

Receiving spam squanders bandwidth; scanning spam uses CPU & memory, storing spam uses hard disk space and I/O.

The measures used to mitigate spam can lead to valid messages being destroyed without indication to the sender or recipient, or overlooked in quarantine folders. Alternatively spams may be bounced back to the purported sender, which is generally not the spammer, so causing backscatter.

What anti-spam measures are used and when can considerably reduce the burden on mail servers – and improve end users experience at the same time.

What can I do about it – and when?

Ideally you control or specify the server where mail for a domain is received: Dealing with spam early increases your arsenal of anti-spam techniques.

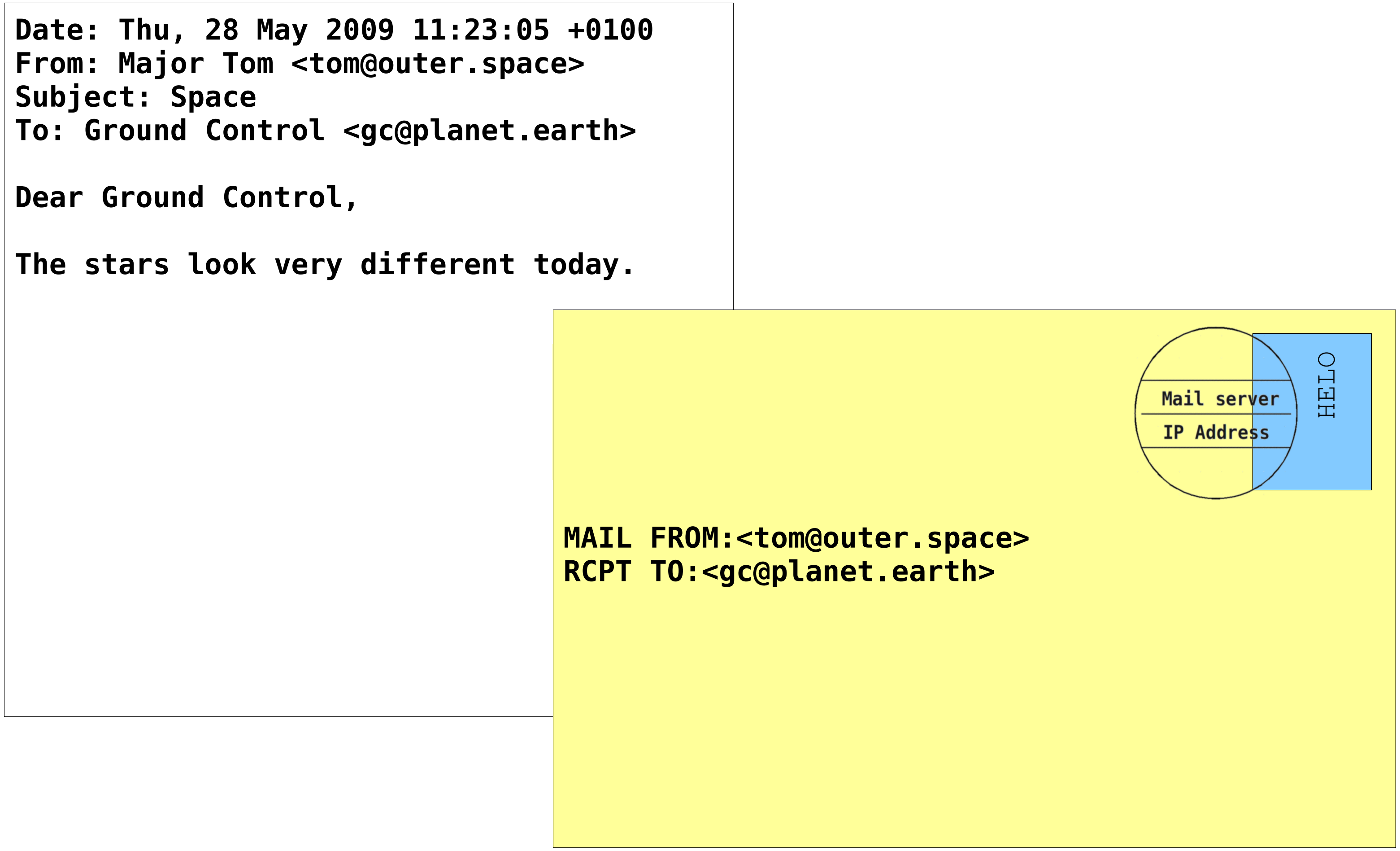
A common misconception is that you have to scan incoming emails to detect spam & viruses.

Prior checks are preferred: The earlier in the SMTP conversation you can decide that a message is junk, the less bandwidth you've wasted.

Scanning is best as a final check: It's complex, it requires CPU, it's imperfect.

Do most of your spam checks before or during the SMTP conversation rather than later. Also gives any senders misclassified as spammers an immediate error and prevents/limits backscatter.

SMTP: Message & Envelope



```
Date: Thu, 28 May 2009 11:23:05 +0100
From: Major Tom <tom@outer.space>
Subject: Space
To: Ground Control <gc@planet.earth>

Dear Ground Control,

The stars look very different today.
```

```
MAIL FROM:<tom@outer.space>
RCPT TO:<gc@planet.earth>
```

SMTP Sample Session

```
(Connect to mail server for the domain planet.earth on port 25)
(Establish TCP connection)
220 mx.planet.earth ESMTP
HELO sputnik.outer.space
250 mx.planet.earth
MAIL FROM:<tom@outer.space>
250 ok
RCPT TO:<gc@planet.earth>
250 ok
DATA
354 go ahead
Date: Thu, 28 May 2009 11:23:05 +0100
From: Major Tom <tom@outer.space>
Subject: Space
To: David <david@planet.earth>

Dear Ground Control,

The stars look very different today.
.
250 ok 1064691857 qp 3569
quit
221 mx.planet.earth
(Connection closes)
```

Receiving system knows the sending IP address,

the stated machine name,

the stated sender,

the destination mailbox,

and the actual message.

Spammers, viruses & SMTP Compliance

Genuine mail servers implement SMTP reasonably carefully. They provide a **quality** service to guarantee delivery for all messages. Speed is a bonus.

Spam sending systems provide **throughput**. They may:

...use dynamic IP addresses

...use RBL-blacklisted IP addresses

...have an invalid HELO/EHLO string

...send commands before being prompted: early talking

...not use standard DNS behaviour to choose a domain's MX

...send from invalid sender addresses

...try to send to invalid addresses

...discard difficult-to-send messages

Connection stage checks

When the remote system tries to connect to your system (minimal bandwidth cost):

- Check the originating IP address: Reverse DNS [exists]
- Check the originating IP on blacklists (RBLs). Choose these carefully
- Check for early talking (low hit rate)
- DNS trick: Dummy low priority MX (= temp fail)
- DNS trick: Nolisting (NB: do it right with no response not temp fail) (I'm still not sure about this)

Prior to SMTP DATA checks

These occur before bandwidth is used for the message contents:

- Check the given HELO / EHLO string: Sensible, poss. also resolvable. Risky.
- Check the sending machine's behaviour, e.g. early talker check
- Greylisting; note possible delay to genuine emails. Also whitelist some domains to avoid unnecessary delays, e.g. list from dnswl.org.
- Is the sender address sensible? (beware null sender valid for bounces)
- Null sender = bounce, so is it to a single recipient?
- Is the recipient address valid? (prevent backscatter)
- Perhaps check target mailbox's quota etc. to ensure deliverability
- SPF available – but imperfect system

Prior to accepting message checks

After receiving the DATA but before accepting the message – these checks are more resource-intensive:

- DKIM check
- VBR check (still being standardised)
- Scan the message for spam: Block 'definite' spam (note: not fully setup in following example thus low hit rate)
- Scan the message for viruses

SMTP-time rejection vs. bounces vs. quarantine

SMTP-time rejection is arguably best where possible:

- Sending system still connected so spoofed sender address doesn't cause backscatter
- A false positive results in a meaningful error to the sender (if human)
- Message or corresponding bounce doesn't have to be queued or quarantined if clearly spam
- Rejection based just on the connection or envelope information means the actual message doesn't need to be received, which would waste bandwidth
- Receiving system may be able to adjust its rules dynamically to lock out a prolific spammer

Reject on failure of specific checks or by scoring on checks and rejecting above a certain threshold.

Putting it all together: In figures

Connection time checks	No. msgs/conns.	% msgs/conns.
No reverse DNS	37750	36.9%
Listed in chosen RBLs	59305	58%

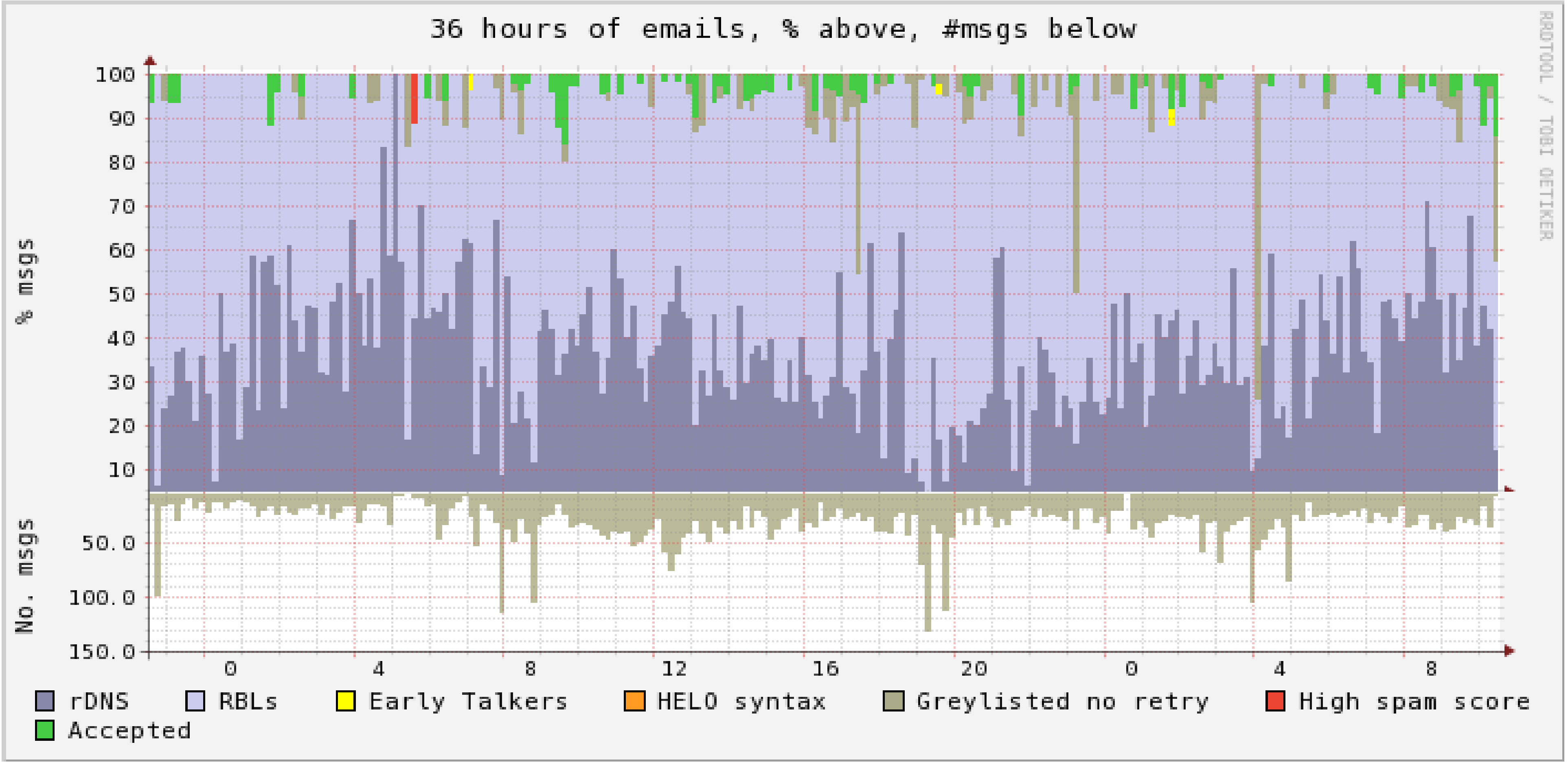
(leaves 5271/5.1%)

Prior to DATA stage	No. msgs/conns.	% msgs/conns.	Of the 5.1%
Early talkers	29	0.03%	0.55%
Invalid HELO/EHLO	14	0.01%	0.27%
Greylisted: Didn't resend	2472	2.42%	46.9%

Prior to accepting message	No. msgs/conns.	% msgs/conns.	Of the 5.1%
Spam check (on fraction)	143	0.14%	2.71%
Email accepted	2613	2.55%	49.57%

Putting it all together: Graph

Here's some data from a small mail system showing SMTP-time rejection:



Testing your own system

Check what measures you already have by spoofing an SMTP session to your server,

- Connect from an IP address with no reverse DNS entry to port 25
- Connect from an RBL-listed address (most domestic DSL IP addresses should be RBL-listed for having dynamic IP addresses) – you can look up your address on RBL's websites
- Use a non-FQDN HELO or EHLO greeting
- Try specifying an invalid recipient at a domain your system controls
- Spoof from an IP address that doesn't normally send email (so not greylisted) and see if you're asked to try again later
- Try sending a message including the GTUBE string
- Try sending a message including the EICAR string

Real-world systems

A few pointers for rDNS, RBL, greylisting checks (Caution: I don't know Postfix, Exim, Sendmail well):

- Qmail, netqmail: rDNS check with '=' in tcpserver cdb file; RBL checking with rblsmtpd; various greylisting implementations.
- Postfix: smtpd_client_restrictions – reject_unknown_client & reject_rbl_client. Also see postconf man page. Various greylisting implementations. Also checkout policyd
- Exim: See wiki.exim.org/Verification, .../AclHeloTricks, .../SimpleGreylisting
- Sendmail: Yuck...

Conclusions

- Significantly reduce the spam/virus-scanning load on your system by using 'old-school' SMTP-time checks first
- Optimise SMTP-time measures to provide better information to valid senders/recipients and reduce quarantining/discards
- Minimise wastage of bandwidth by blocking on connection or envelope information
- Lightweight SMTP-level measures scale well; SMTP-time scanning may not
- Time-sensitivity of your business operations may prevent your use of certain tests esp. greylisting, although careful whitelisting may address this.
- Easy to inadvertently block valid email
- A system with a reduced spam burden has spare capacity for other tasks or can reduce its electricity use; improves efficiency for non-spam messages

Questions