Postfix, lessons learned and recent developments
Overview

- Overview.
- Motivation and architecture.
- Spam around the clock.
- Scalable defense (postscreen zombie blocker).
- New: miscellaneous improvements.
- New: security without global PKI (DANE).
- New: replacing Berkeley DB with LMDB.
- Conclusion.
Postfix timeline
Larger is not necessarily better

Most of this presentation

Overview

Time (year)

Line count (K&R, no comments)

Postfix
Sendmail
qmail

Postfix "complete"
Postfix motivation and architecture

Why (not) write another UNIX mail system
CERT/CC advisories for Sendmail
The initial threat model: mail server attacks

<table>
<thead>
<tr>
<th>Advisory</th>
<th>Version</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA-1988-01</td>
<td>5.58</td>
<td>Unprivileged access</td>
</tr>
<tr>
<td>CA-1993-16</td>
<td>8.6.3</td>
<td>Unprivileged access</td>
</tr>
<tr>
<td>CA-1994-12</td>
<td>8.6.7</td>
<td>Full system privilege</td>
</tr>
<tr>
<td>CA-1995-05</td>
<td>8.6.9</td>
<td>Full system privilege</td>
</tr>
<tr>
<td>CA-1995-13</td>
<td>8.7.0</td>
<td>Full system privilege</td>
</tr>
<tr>
<td>CA-1996-04</td>
<td>8.7.3</td>
<td>Full system privilege</td>
</tr>
<tr>
<td>CA-1996-20</td>
<td>8.7.5</td>
<td>Full system privilege</td>
</tr>
<tr>
<td>CA-1996-24</td>
<td>8.8.2</td>
<td>Full system privilege</td>
</tr>
<tr>
<td>CA-1996-25</td>
<td>8.8.3</td>
<td>Group privileges</td>
</tr>
<tr>
<td>CA-1997-05</td>
<td>8.8.4</td>
<td>Full system privilege</td>
</tr>
<tr>
<td>CA-2003-07</td>
<td>8.12.7</td>
<td>Full system privilege</td>
</tr>
<tr>
<td>CA-2003-12</td>
<td>8.12.8</td>
<td>Full system privilege</td>
</tr>
<tr>
<td>CA-2003-25</td>
<td>8.12.9</td>
<td>Full system privilege</td>
</tr>
</tbody>
</table>
Postfix low-privilege architecture
(omitted: non-daemon programs for submission and management)

Motivation & architecture
Spam around the clock
SPAM is a 24-hour operation ...


Spam around the clock
... but many spambots are not


Spam around the clock
Spam connections/hour at charite.de (time in UTC)
From IP addresses blacklisted at zen.spamhaus.org, Oct 29 – Jan 23, 2014

Spam around the clock
Zombies suck the life out of the mail server

Adapting to changing threats
Email spam percentage over time (Symantec)
August 2010: 92% Of email is spam, 95% of spam is from botnets
postscreen zombie blocker
Prior work: OpenBSD spamd, MailChains TrafficControl, M.Tokarev

Light-weight screening process

Postfix default: 100 sessions

Scalable defense
postscreen – the first step in a four-layer defense

- **Postscreen connection triage**: Block spambots that send ~90% of all SPAM
- **Postfix SMTP server**: Block SPAM with SMTP-level access policies
- **Header/body regular repressions**: Block SPAM/backscatter with simple signatures
- **Deep content inspection**: Block remaining SPAM with SpamAssassin etc.

Scalable defense
postscreen workflow - tests before SMTP handshake
One daemon screens multiple connections simultaneously

Fast path: ~0.1 ms
Accept connection

Is client in temp whitelist?
Yes
Hand-off to real SMTP server

No
Static W/B list
DNS W/B list
Pregreet test
(Primary MX test)

Slow path: up to ~6 sec
Add client to temp whitelist
Pass

Fail
Reject or defer mail
(and log from, to, client, helo)

Close connection

dummy SMTP and TLS engine

Scalable defense
postscreen DNSBL/DNSWL support
Parallel DNS lookups

- Weight factors (to whitelist, use negative numbers).

```plaintext
postscreen_dnsbl_sites = zen.spamhaus.org*2, bl.spamcop.net*1, b.barracudacentral.org*1
postscreen_dnsbl_threshold = 2
```

- Reply filters.

```plaintext
postscreen_dnsbl_sites = zen.spamhaus.org=127.0.0.4 ...
postscreen_dnsbl_sites = zen.spamhaus.org=127.0.0.[1..11] ...
```

- Allow “good” clients to skip all other tests.

```plaintext
postscreen_dnsbl_sites = list.dnswl.org=127.0.[0..255].[1..3]*-2 ...
postscreen_dnsbl_whitelist_threshold = -2
```

Scalable defense
Making zombies bark - multi-line greeting trap

- Good clients wait for the full multi-line server greeting:

  ```
  postscreen: 220–server.example.com ESMTP Postfix<CR><LF>
  smtp server: 220 server.example.com ESMTP Postfix<CR><LF>
  good client: HELO client.example.org<CR><LF>
  ```

- Many spambots talk immediately after the first line of the multi-line server greeting:

  ```
  postscreen: 220–server.example.com ESMTP Postfix<CR><LF>
  postscreen: (wait a few seconds)
  spambot: HELO i-am-a-bot<CR><LF>
  ```
Over 60% of bots pregreet (8% not on DNSBL)

mail.charite.de, Berlin, Aug 26 – Sep 29, 2010

Scalable defense
New developments: miscellaneous improvements
Miscellaneous Postfix 2.11 improvements

- Documentation: “Perfect” Forward Secrecy.
- TLS-encrypted MySQL database connections.
- Both “user+suffix@example” and “user–suffix@example”.
- Managing master.cf files without text editor (just like main.cf).
  - Primary target: third-party management tools.
  - Basic idea: everything is a “name = value” pair:
    - `postconf -F "*/*/chroot = n"`
      - Sets the “chroot” field to “n” for all master.cf entries.
    - `postconf -P "smtp/unix/smtp_bind_address = 192.0.2.1"`
      - Sets “-o smtp_bind_address=192.0.2.1” on “smtp unix ...” master.cf entry.
New developments: security without global PKI

DNS-based authentication of named entities (DANE)
Global PKI violates the principle of least privilege

- Hundreds of root CA certificates (Windows ~350, IOS ~200).
  - Owned by ~100 distinct organizations world-wide.
  - Hundreds (or more) registration authorities (RAs) world-wide.

Security without global PKI
SMTP over TLS – no server certificate verification
RFC 3207, published 2002

- Problem: RFC does not require certificate name verification.
  - Why not the recipient domain name (example.com below)?
    • One mail server may host many domains (RFC predates SNI).
  - Why not the mail server hostname (lb-01.spam.filter below)?
    • The mail server hostname is looked up with insecure DNS.

Simplified connection setup procedure

| DNS query: example.com MX?          |
| DNS reply: example.com MX lb-01.spam.filter |
| DNS query: lb-01.spam.filter A?     |
| DNS reply: lb-01.spam.filter A 192.0.2.1 |
| Negotiate TLS with host = 192.0.2.1, port = 25 |

Security without global PKI
SMTP over TLS – downgrade vulnerability

- Problem: the client doesn’t know that it should use TLS.
  - `<user@example.com>`, not `<smtps://user@example.com>`.
  - Plaintext is the default, TLS is optional.

<table>
<thead>
<tr>
<th>No downgrade attack</th>
<th>With man-in-the-middle downgrade attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>S: 220 server.example.com</td>
<td>S: 220 server.example.com</td>
</tr>
<tr>
<td>C: EHLO client.example.org</td>
<td>C: EHLO client.example.org</td>
</tr>
<tr>
<td>S: 250-server.example.com</td>
<td>S: 250 server.example.com (No STARTTLS announcement)</td>
</tr>
<tr>
<td>250 STARTTLS</td>
<td>C: MAIL FROM:<a href="mailto:user@example.org">user@example.org</a></td>
</tr>
<tr>
<td>C: STARTTLS</td>
<td>S: 250 Sender address accepted</td>
</tr>
<tr>
<td>S: 220 Ready to start TLS</td>
<td><strong>Plaintext throughout the entire session</strong></td>
</tr>
<tr>
<td><strong>No plaintext from here on</strong></td>
<td></td>
</tr>
</tbody>
</table>

Security without global PKI
RFC 6698 (DANE) to the rescue

DNS-Based Authentication of Named Entities

- Introduces TLSA¹ DNS records with:
  - Expected server (or issuer) certificate or public key.
  - Or better: their SHA-256 or SHA-512 hash.

- Requires secure DNS (DNSSec).
  - Unavoidable when using DNS for secure authentication.

¹RFC 6698: "TLSA does not stand for anything".  

Security without global PKI
Two preferred (SMTP) DANE deployments

- **Mini PKI**
  - DNS TLSA hash of issuer certificate or issuer public key
  - Must match the server hostname

- **No PKI**
  - DNS TLSA: hash of server certificate or server public key

Security without global PKI
Concrete example with debian.org
Not showing the DNSSEC signature records (RRSIG)

- Look up the debian.org mail server names:
  Reply: debian.org MX 0 mailly.debian.org
debian.org MX 0 muffat.debian.org

- Look up mailly A records:
  Reply: mailly.debian.org A 82.195.75.114

- Connect to 82.195.75.114 port 25.

- Look up mailly TLSA records:
  Reply: _25._tcp.mailly.debian.org TLSA 3 1 1 [SHA-256 of TLS server public key]

- Match TLSA record with SHA-256 of TLS server public key.
  Security without global PKI
Securing SMTP with DNSSEC and DANE

- Minimized trust.
  - Not: 100s of RAs.
  - Secure copy of root zone public keys.
  - DNS target zone plus its ancestors.
    - Maybe: issuer cert.
- No downgrade attack.
  - Use TLS when DNS TLSA record exists.

Security without global PKI
DANE support in Postfix 2.11 stable release

- Aug 2012: RFC 6698 is published.
- Q1 2013: Start of Postfix implementation (Viktor Dukhovni).
- Jan 2014: DANE support in Postfix stable release.
  - Requires DNSSec validating resolver (e.g., BIND or unbound).
- Please try DANE support, but be prepared for surprises.
  - A few DNS servers mis-handle TLSA queries.
    - Use “dane enabled” as default.
    - Use “dane disabled” SMTP TLS policy for problem sites.
    - See TLS_README (or the upcoming DANE_README).

Security without global PKI
New developments: LMDB database support

Unintended consequences of adopting AGPL
June 2013: Oracle updates Berkeley DB 6.0 license
Popular open-source key-value store

- Berkeley DB v5: two licenses, copyleft and commercial.
  
  Copyleft: make all source code available if you *distribute work* that uses Berkeley DB.

- Berkeley DB v6: two licenses, AGPLv3\(^1\) and commercial.
  
  AGPL: also make all source code available if you *provide network service* that uses Berkeley DB.

- Problem: cannot legally combine GPLv2 and AGPLv3 code without relicensing the GPLv2 code (AGPLv3 would be OK).

\(^1\)GNU Affero General Public License version 3.0. Pronunciation: /ˈaf.ə.roː/.
LMDB - Lightning Memory-Mapped Database
Author: Howard Chu

- Described by some as a Berkeley DB replacement.
- OpenLDAP Public License.
- Memory-mapped, max size limited by memory address range (typically ~31 bits on i386, ~47 bits on x86_64 or ~128 TB).
- Copy-on-write, zero-copy, MVCC, multi-reader, single-writer.
- Ported by its author to dozen+ other open source projects.
- Postfix integration took 5 iterations with changes to both Postfix and LMDB.

Replacing Berkeley DB
Challenges integrating LMDB into Postfix

- Hard database size limit, specified when database is opened.
  - Postfix processes fail unexpectedly if size limit is set too low.
  - LMDB 0.9.8 allows Postfix to resize database on the fly.

- LMDB lockfile must be writable by readers. Hard limit on number of readers, specified when database is opened.
  - World-writable lock files, for example under /etc/postfix.
  - Postfix process fail unexpectedly if reader limit is set too low.
  - LMDB 0.9.9 allows Postfix to use external (fcntl()-based) locks.

Replacing Berkeley DB
Challenges integrating LMDB into Postfix

- Information leak: writing ~4kbyte chunks of uninitialized heap memory to the LMDB database file.
  - Contains traces from past activity in the same process, not necessarily meant to persisted or shared.
  - LMDB 0.9.10 initializes malloc()ed memory by default.

- LMDB library functions rely on assert() extensively.
  - Write a message to stderr and abort the program immediately.
  - Postfix daemons fall out of the sky without logging any error.
  - LMDB 0.9.11 allows Postfix to log an error message.

Replacing Berkeley DB
LMDB support in Postfix 2.11 stable release

- First persistent Postfix database that safely supports multiple writers such as postscreen.
- Not exactly a Berkeley DB drop-in replacement – requires additional Postfix code to recover from “hard limit” errors.
- Expect better safety than Berkeley DB, mainly due to COW.
Postfix lessons learned

- Invent sparingly: don’t re-invent what works.
  - SMTP, Milter, maildir, Sendmail lookup tables.

- Build the stable protocols into Postfix.
  - SMTP, LMTP, TLS, SASL, IPv6, DSN, MIME, LDAP, SQL, CDB, memcache, LMDB, (DANE).

- Plan for change: provide safe plug-in interfaces for future proofing.
  - Anti-Spam, Anti-Virus, DKIM, SenderID, SPF, greylist.

- Optimize both worst cases and common cases.
  - On the Internet, worst cases will become common cases.

- Don’t let a C prototype become the final implementation.

Conclusion