Mail Transfer 2 (MT2)

http://www.ntrg.com/specs/mt2/mt2-12.ppt

Eric A. Hall
ehall@ntrg.com

June 24, 2003

Minimalist Architecture

- Layered data model
- End-to-end extensions
- Validated host/user identity information
- Protocol semantics
- Routing services
- Gateway definitions

Everything else is a layered service

Layered Data Model

- Three MIME objects for each message:
  - message/envelope
  - message/headers
  - message body (text/plain, etc)
- Each message object is transferred and stored separately

Message Data Objects

- Message Envelope and Headers use XML
  - Structured header field entities
  - Registered namespace of entities
  - Direct support for UTF-8
- Message Body is raw MIME entity
  - Text/Plain, etc.
  - Eight-bit data-widths explicitly allowed

Message Envelope

- Contains data that is not directly related to the message body
  - Envelope senders/recipients
  - Signatures/certificates
  - Transfer-path trace data
  - Delivery extensions (e-postage tags, etc.)
  - Error report meta-data

Message Headers and Body

- Message headers contain data related to the message, but not part of the contents
  - To, From, Subject, ...
  - Recipient extensions (vCard requests, etc.)
- Message body contains payload
  - Traditional email messages
  - Hinting data for control messages
- Both can be signed/encrypted

Message Transfer and Storage

- Each object transferred separately
  - Explicit permission-to-send for each
  - Explicit acknowledgement for each
  - Allows pre-transfer filtering/rejection
- Each section must be accessible discretely
  - Encrypted message headers need separate data
  - Control messages might have envelope only
  - IMAP header fetching

End-to-End Extensions

- Different extension types
  - Transfer extensions give hop-by-hop features
  - Delivery extensions give @domain features
  - Recipient extensions give user@ features
- Extensions provide carrier service
  - whitelist negotiation, hashcash and e-postage fulfillment, spam-trap matches, mailing list management, chess-by-mail, etc.

Extension Architecture

- Only a few extensions should be defined in the core spec
  - Necessary negotiation controls and errors
  - Rest pushed out to layered service space
- Extending protocol and agents must be easy
  - Installing application should ~transparently enable the associated options (not too easy!)
  - OID-schema registries for automation?

Transfer Extensions (cont’d)

- Hop-by-Hop feature negotiation
- Extension-specific verbs or OPT parameters
  - Message status (STAT command)
  - RFC1122 TCP Urgent (OPT URG parameter)
- Extensions have private OID branches
  - Extension namespace uses OID space
  - May reuse standard return codes
- Errors returned in-band

Delivery Extensions

- Processed by @domain after last-hop
  - Hashcash/e-postage payment data
  - Negotiated whitelists
  - Delivery notifications
- Stored in message/envelope object
- Errors returned via control messages

Delivery Extensions (cont’d)

- Carried in message/envelope
  - Unencrypted
  - Extensions can be flagged as critical
  - Unknown critical extensions cause entire message to be rejected
- Extensions have private OID branches
  - Extension namespace uses OID space
  - May reuse standard return codes

Recipient Extensions

- Processed by localpart@
  - Request your vCard
  - Out-of-band application control messages (eg, mailing list management)
  - “Black queen takes rook”
  - Disposition notifications

Recipient Extensions (cont’d)

- Carried in message/headers
  - Can be encrypted/signed
- User must control default processing
  - Automatic processing allows worms
  - Automatic errors are silent notifications
- Extensions have private OID branches
  - Extension namespace uses OID space
  - May reuse standard return codes

Validated Identities

- Hosts and sender certificates
  - Host identity presented during session setup
  - Transfer headers recursively signed with
    - Sender identity
    - Per-hop host identity
- Identity can be used for several filters
  - Access-control restrictions
  - Extension restrictions

Identity Types

- User identity bound to email address
  - Personal description (name, etc.) cannot be trusted unless CA is trusted
- Host identity bound to hostname
  - Hosts also have process@domain user certificates for error messages, control messages, etc.

Validity vs Trust

- Validation only speaks to authorization
  - The parties can be verified as authorized to use the certificates that they present
  - Broad enforcement is possible at this level
- Validation does NOT speak to trust
  - Does NOT ensure they are who they say (only that they are authorized to say it)
  - 3rd-party “vouch lists” needed for trust

Validation Mechanisms

- Three allowable validation mechanisms
  - If issuer is known/trusted, MUST validate against local CA certificate repository
  - If issuer is not known/trusted, MAY validate against public delegation data (see next slide)
  - Hosts can explicitly trust another host to have performed validation (eg, interior gateway)
- Private links can exchange private CA certs

Trust Mechanisms

- No mandatory trust mechanisms
  - Trusted certificate authorities, presumed to have verified identity information
  - Commercial trust-broker lists, eg bonded senders and other whitelists
  - Community trust-broker lists, eg public trust lists, blacklists of known abusers

Delegated CA Validation

1) Extract issuerAltName dNSName attribute from user/host certificate
2) Verify that the issuer domain name is a delegation parent of the subject name
3) Lookup issuer domain name and retrieve certificate data
4) Validate the host/user certificate

FIRS (CRISP WG) Sample

- Sample host certificate:
  - Subject: goose.ehsco.com
  - Alt Issuer: ehsco.com (path to subject is good)
- Generate LDAP lookup
  - Srchbase: cn=inetResources,dc=ehsco,dc=com
  - Assertion: (&(objectclass=inetDnsDomain) (dn:cn:=ehsco.com))
  - Attribute: caCertificate

Fungible Private CA Certificates

- Parties can change CA certificates at will
  - Doing so will invalidate all previously issued host/user certificates
  - Admins can still filter against the domain name in the issuerAltName field and preempt all user/host certificates from that issuer
  - 3rd-parties can offer issuerAltName blacklists
- 3rd-parties can offer vouch lists for orgs, adding extra credibility

Protocol Semantics

- Stateful sessions
  - Session setup
  - Message transfer loop (repeat as needed)
  - Session teardown
- Asynchronous within each state
  - Interleaved data and commands/responses
  - Full-duplex on-demand (no TURN)
  - TCP Urgent allows commingling

Request Semantics

- Each request provides:
  - Sequence number tag for the request
  - Verb for the request
  - Verb-specific parameters
  - Extensions and parameters enclosed in ( ) pair
  - Full request enclosed in [ ] pair
- Simple operations use one transaction pair, while data-transfer operations use two pairs

Standard Command Verbs

- HELO {cert-size} (send host identity)
- OPT <extension=parameters> <...>
  - PIPE=ON (enable/disable pipelining)
  - TRACE=ON (enable/disable traceroute)
  - URG (RFC1122 TCP Urgent compliance)
- XFER msg-id MIME/type {part-size} {num-parts}
- ABOR tag (kill previous command)
- NOOP (keep-alive)
- QUIT

Response Semantics

- Each response provides:
  - Original sequence number tag
  - Static command response codes (OK/ERR/...)
  - Extensible command result codes (OID.n.n)
  - Free-text message
  - Extension responses enclosed in ( ) pair
  - Full response enclosed in [ ] pair
- Unsolicited responses use “*” for tag

Response and Result Codes

- Response codes indicate acceptance
  - OK, command accepted and processed
  - ERR, command refused or fatal failure
  - TMP, command pending additional input
- Result codes provide detailed output
  - Standard and extension-specific OIDs
  - OID codes are extensible, no collisions

Request/Response Sample

- Syntax example

```
C:[seq verb <params> <(ext1 <params>)> <...>]
S:[seq rsp ret <(ext1 ret text)> <...> text]
```

- OPT negotiation non-normative sample

```
C:[1 OPT (BAZ=FOO;BAR)]
S:[1 ERR 99.0.5.0 (BAZ ERR 99.0.5.9 Unknown.)]
```

- Data-transfer non-normative sample

```
C:[2 XFER m23@test.com message/envelope {3279} {0}]
S:[2 TMP 99.0.3.5 Go ahead with envelope.]
C:(3279 octets)
S:[2 ERR 99.0.5.23 Invalid sender certificate.]
```

Session Setup

- Server sends list of anonymous options
- Client sends greeting command and data
  - Client sends host certificate
  - Server validates and checks permissions
  - Server may send its own host certificate
  - Systems may negotiate encryption
- Server sends list of authenticated options

Setup Example

- Non-normative greeting sample

```
S:[* TMP 99.0.3.1 (VER=0.9;1.0) Hi there.]
C:[1 HELO {4928}]
S:[1 TMP 99.0.3.2 Friend or foe!]
C:(4928 bytes of certificate data)
  <optional bilateral exchange, encryption>
S:[1 OK 99.0.2.1 (BAR) Greetings friend.]
C:[2 OPT (BAR=FOO)]
S:[2 OK 99.0.2.2 (BAR OK 121.0.2.1) Go ahead.]
```

Transfer Semantics

- Three steps to each message transfer
  - – message/envelope
  - – message/headers
  - – Message body (MIME body)
- Each step has two transaction pairs
  - – Request to send
  - – Actual send
- Untrusted hosts may be forced synchronous

Transfer Semantics (cont'd)

- Request-to-send parameters:
  - – Message-ID for the message
  - – Message part (envelope/headers/body)
  - – Size in octets of part fragment
  - – Number of pending fragments
- Server responses for each pair
  - – Explicit permission to send the data
  - – Acknowledgement for the actual data

Transfer Example

- Non-normative transfer sample

```
C:[2 XFER m23@test.com message/envelope {3279} {0}]
S:[2 TMP 99.0.3.5 Go ahead with envelope.]
C:(3279 octets)
S:[2 ERR 99.0.5.23 Invalid sender certificate.]
```

- Non-normative abort sample

```
C:[3 XFER m23@test.com message/body {59203} {0}]
S:[3 TMP 99.0.3.8 Go ahead with message body.]
C:(only 200 octets)
C:[4 ABOR=3]
S:[4 OK 108.0.2.1 Command number 3 killed.]
S:[3 ERR 99.0.5.23 Transfer failed.]
```

Performance Characteristics

- Unknown/untrusted entities treated warily
  - – Half-duplex, synchronous transfers
- Faster-than-NNTP bulk transfers for known and trusted entities
  - – Interleaved, asynchronous data objects and command/response pairs
  - – Post-transfer delayed rejections
  - – Nailed-up full-duplex sessions

Anti-Spam Capabilities

- Pre-transfer filtering
  - – Host untrusted, message too large, etc.
  - – Trust problems
- Post-transfer filtering
  - – Delivery extension filtering
  - – Recipient extension filtering
  - – Extensible architecture

Pre-Transfer Filtering

- Invalid host certificate
- Hostname/subject mismatches
- Unauthorized client
- 3rd-party host/domain black/whitelists
- Invalid trace-data
- Prohibited senders, recipients, subject...
- Encryption levels
- Undesirable options
- MIME syntax errors
- Prohibited content

Pre-transfer filtering saves $

Post-Transfer Filtering

- Delivery extensions
  - – Can be enabled per-domain or per-recipient
  - – E-Postage payment data
  - – Hash-cash proof data
- Recipient extensions
  - – Challenge-response proof data
  - – “Not-in-address-book” vCard negotiation

Routing Services

- @domain routing
  - – Public routing with DNS SRV?
  - – Public routing with CRISP extension?
  - – Private routing with other services?
  - – Define weighting metrics
- recipient@ routing is possible
- Extension-specific routing is possible

Legacy Messaging Integration

- Bi-directional object mappings
  - – Message parts mapped to MIME entities
- Bi-directional header mappings
  - – “Received” mapped to “<RFC821.Received>”
- Identity mappings
  - – SMTP sender mapped to sender certificate
- Must be reversible both ways

Deployment Staging

- Site-to-Site transfers for carriers
  - – Email houses
  - – Large-scale ISPs
- Site-to-Site for medium/small shops
- Extend to clients eventually
- Will need upgrades to POP/IMAP as well for full end-to-end capabilities

Summary

- Host/user/domain identity information
  - – Filtering
  - – Private enforcement actions
- End-to-end extensions
  - – Anti-spam delivery applications
  - – User-based extra-mail applications
- Performance enhancements
- Can be deployed relatively quickly