Sendmail Security

Gregory Neil Shapiro
Sendmail, Inc.

<gshapiro@sendmail.org>

PGP Fingerprints:
RSA: 66 39 58 9A 83 5F 52 26 88 E4 59 36 5A 94 D9 48
DSS: 6990 834A 0D85 C4D7 D0B5 3891 18F5 0380 24C2 7802

Bugs/Questions to: <sendmail@sendmail.org>

Copyright © 2001 by Eric P. Allman
and Gregory Neil Shapiro
All Rights Reserved
Outline

♦ Overview and introduction

♦ Securing sendmail
  ⇒ Why does sendmail need special privileges?
  ⇒ Running without set-user-ID root
  ⇒ Options to reduce exposure
  ⇒ Mail system security
  ⇒ Denial of service attacks

♦ Securing the mail environment
  ⇒ Controlling relaying
  ⇒ Privacy
    • SMTP authentication
    • STARTTLS
  ⇒ Logging and tracing

♦ System examples

♦ Parting advice
Overview and Meta-Information

♦ Talk based on sendmail 8.12
  ⇒ New features marked in margin (8.12)

♦ Interactive — please ask questions

♦ References (section/page) at bottom of page are to sendmail (O’Reilly & Associates, 2nd edition)
  ⇒ Unfortunately, book documents 8.8

♦ Assumptions
  ⇒ You know how sendmail works in general
  ⇒ I will point at features, but you will have to read for the details
    • doc/op/op.ps
    • cf/README
    • sendmail/README
    • sendmail/SECURITY (8.12)
What Do We Mean By “Security”?  

◆ “Is sendmail secure?”
  ⇒ Should not be able to get special permissions via the mail system
  • root or any other user
  • Execute, write, read
  ⇒ Denial of service should be “hard”
  • Never really impossible: manage rather than prevent
  • Should degrade gracefully

◆ “Is my mail secure?”
  ⇒ Avoid information leakage
  • In transit
  • On disk
  ⇒ Prevent theft of service
  • Spam
  • Third party relaying
  ⇒ Make forgery as hard as possible
General Security Principles

♦ Minimal permissions and services
♦ Tradeoff between usability and security
♦ Things to tune in sendmail:
  ⇒ Ownerships
  ⇒ File and directory modes
  ⇒ Option settings
♦ As of 8.9, sendmail is paranoid by default
  ⇒ Gets even more paranoid in newer versions
What Does Sendmail Need root For?

♦ Access to files
  ⇒ Reading :include: files
  ⇒ Reading forward files
  ⇒ Reading and writing queue files
  ⇒ Reading SMTP AUTH password files  (8.10)
  ⇒ Access to private maps

♦ Sockets
  ⇒ Binding to low port numbers
    • Ports 25 and 587 for SMTP server
    • May need to rebind after startup

♦ Running as a different user
  ⇒ Delivery to files
    • In aliases, forward, and :include: files
  ⇒ Delivery to programs
    • Local delivery agent (LDA) (e.g., mail.local)
    • |program in aliases, forward, and :include: files
No More Set-user-ID root (8.12)

♦ Single binary, two modes of operation
  ⇒ Mail Transfer Agent (MTA)
  ⇒ Mail Submission Program (MSP)
  ⇒ Requires new user and group:
    smmsp:smmsp
      • User and group should not be used for anything else
      • sendmail binary is set-group-ID smmsp

♦ Decides mode based on command line flags
  ⇒ sendmail -bm (default), -bs, and -t use MSP mode
  ⇒ Otherwise assume MTA mode
  ⇒ Can override using -Ac (MSP mode) or -Am (MTA mode) flags

♦ Each mode has its own configuration file
♦ Can emulate 8.12 separation in older sendmail versions
♦ See sendmail/SECURITY for more information
Mail Transfer Agent (MTA)

♦ Uses `/etc/mail/sendmail.cf`
♦ Started by `root` ⇒ Usually at boot time
♦ Listens on network ports
♦ Delivers mail
♦ In other words, no change from previous versions
♦ Tip: Start with `-L sm-mta` to distinguish MTA and MSP mail logs (8.10)
♦ Gotchas (8.12)
  ⇒ Only `root` can run `mailq`
  ⇒ `sendmail -bv` may give misleading output for normal users
Mail Submission Program (MSP)  (8.12)

♦ Uses /etc/mail/submit.cf
♦ Reads mail from standard input
♦ Passes everything to MTA
  ⇒ No local delivery or address expansion

♦ Uses set-group-ID to write to a separate queue
  ⇒ Allowed due to UseMSP option in submit.cf
  ⇒ Default: /var/spool/clientmqueue/
    • Must be owned by smmsp:smmsp and be group writable
      » Not world writable
  ⇒ Should run queue periodically:
    sendmail -L sm-msp-queue -Ac -q30m
    • Start as root
    • Runs as smmsp due to RunAsUser setting
    • Alternatively, run client queue via cron
    • For the paranoid, can also start as smmsp:
      echo "sendmail -L sm-msp-queue -Ac -q30m" | su smmsp
Users special to sendmail

- **DefaultUser**
  - Default user ID for running programs
  - Defaults to `mailnull` if in `/etc/passwd`, else daemon, else `sendmail`, else uid:gid of `1:1` (8.9)

- **TrustedUser** (8.10)
  - User trusted to own maps, aliases, and other files and start daemon
  - Set with great care
  - Defaults to `root`

- **RunAsUser**
  - User that sendmail becomes after accepting a connection in daemon or after initializing if not a daemon
  - Defaults to `root`
  - If non-`root`, most other rules change
RunAsUser Option

♦ sendmail gives up root privileges after SMTP connection is accepted
♦ Cannot become another user, ever
♦ Mail queue must be owned by RunAsUser
♦ All maps must be readable by RunAsUser
♦ All :include: files must be readable by RunAsUser
♦ All files will be written by RunAsUser†
♦ All programs will be invoked by RunAsUser†
  ⇒ † In 8.9; beginning with 8.10, these addresses are disallowed unless the DontBlameSendmail option includes NonRootSafeAddr (8.10)
  • DontBlameSendmail=NonRootSafeAddr
Tuning File Paths and Permissions

♦ **ForwardPath**

⇒ If possible, limit locations or do not use (tips later)

♦ **QueueFileMode** \((8.12)\)

⇒ Permissions for queue files, set with care
⇒ Should be 0600 for MTA, 0660 for MSP

♦ **TempFileMode**

⇒ Permissions for other files written by sendmail, set with care
  • Delivery to files
  • Host status files
  • Transcript files
  • Dead letter files
  • Queue files \((pre-8.12)\)

REF: 22.8.3.2/351; 34.8.27/710–1; 13.2.4/143; 22.8.3.7/353; 34.8.68/755
SafeFileEnvironment Option

♦ Limits files written to specified path
  ⇒ User mailboxes only — does not apply to aliases, maps, etc.
  ⇒ Does not apply to files written by delivery agents (e.g., /bin/mail)

♦ Uses chroot(2) to avoid mistakes

♦ Restricts writes to be regular files (no devices)

♦ Initial path optional on user file name
  ⇒ Assume SafeFileEnvironment=/safe
  ⇒ /safe/file/path writes /safe/file/path
  ⇒ /file/path writes /safe/file/path
General Things to Secure

♦ Check file and directory modes
♦ Check aliases file for bogus entries
♦ Create mailnull account in /etc/passwd

⇒ No valid password (e.g., ‘*’)
⇒ No valid shell
⇒ No files owned by this user

♦ Use smrsh program to limit choice of programs that can be executed in forward and aliases files
♦ If necessary and possible, consider running in chroot area or FreeBSD jail
Mail queue has root impact information
⇒ /var/spool/mqueue —0700 RunAsUser
⇒ /var/spool/clientmqueue —0770 smmsp
(8.12)
⇒ /, /var, /var/spool —0755 root

/etc has root impact information
⇒ /etc —0755 root

Mail configuration has root impact information, but needs to be modifiable by TrustedUser
⇒ /etc/mail —0755 TrustedUser
  • Put all maps here
⇒ /etc/mail/* —0644 TrustedUser
  • Except private files
    » STARTTLS server and client key
    » Authentication info file or database

If absolutely necessary, can adjust sendmail paranoia with DontBlameSendmail
Restrictive File Access

♦ Some in 8.8; stronger in 8.9 and later
♦ Do not read files that are group or other writable
  ⇒ Attack: change victim’s writable forward file to overwrite victim’s private file; mail victim
♦ Do not read files in group or other writable directories
  ⇒ Attack: similar (create new file, give it away)
♦ Do not read forward files that are links
♦ Do not write files in group or other writable directories
  ⇒ Attack: symlink $aliases.db$ to $/etc/passwd.db$, force aliases rebuild
♦ Many others
♦ Can be turned off using **DontBlameSendmail** option
  ⇒ Use with caution
Clean Aliases File

♦ Control the aliases file

♦ Never alias to programs that trust their input
  ⇒ Famous case: uudecode

♦ Should never be writable by non-root user
  ⇒ Actually, can be writable by TrustedUser(8.10)

♦ Directory should also be protected
  ⇒ Attack: move real file aside, replace with bogus version
**smrsh**

- Used in place of `/bin/sh` in `sendmail.cf`
  
  ⇒ Get this using `FEATURE('smrsh')`

- Limits programs to those in a specified directory
  
  ⇒ Of course, this directory should be writable only by `root`
  
  ⇒ Avoids mistakes (or worse) by users
  
  ⇒ User-specified paths stripped off
  
  ⇒ Shell meta-characters restricted

- **Examples**
  
  ⇒ “/usr/ucb/vacation eric” executes “/usr/adm/sm.bin/vacation eric”
  
  ⇒ “cat /etc/passwd” is rejected (cat not in `/usr/adm/sm.bin`)
  
  ⇒ “vacation eric < /etc/passwd” is rejected (redirection not allowed)

- Do not allow `procmail` or any other program which can run arbitrary programs!
Other Tricks for the Paranoid

♦ Run mailers in a *chroot*ed directory  
  ⇒ Use /= equate in mailer definition

♦ Do not permit normal users to see the mail queue  
  ⇒ Avoids traffic analysis  
  ⇒ *PrivacyOptions=restrictmailq,restrictqrun*  
  ⇒ Only people in same group as the queue directory can see the queue  
  ⇒ Only *root* or owner of the queue directory can run the queue  
  ⇒ Be sure to turn off read permissions on the logs
Running with *chroot* or *jail*

- Only protects daemon, useful for firewalls
- If using *chroot*, useless if **RunAsUser** is not set
- Need to create minimal filesystem in secure area
  - Documented in *jail*(8) man page
  - For *chroot*, varies depending on architecture

```
/safe/dev/null
/safe/etc/{group,passwd,resolve.conf}
/safe/etc/mail/*
/safe/usr/lib/lib*.so.*
/safe/usr/libexec/ld.so
/safe/var/spool/mqueue
/safe/usr/sbin/sendmail
```
- May need to experiment to get this right

- *sendmail* binary should not be set-user-ID **root**
- Start sendmail inside secure area:

```
chroot /safe /usr/sbin/sendmail -bd -q30m
jail /safe host IP /usr/sbin/sendmail -bd -q30m
```
Denial of Service Attacks

♦ Generally, anything that prevents others from getting mail through
♦ Shutting down SMTP port
♦ Flooding queue disk partition
♦ Clogging outgoing connections
♦ Filling the process table
♦ Many others
Avoiding Denial of Service Attacks

♦ Generally, can not stop them
♦ Example: avoid flooding host with tons of sendmail processes
  ⇒ Set MaxDaemonChildren
  ⇒ BUT... this allows trivial shutdown of SMTP port
♦ Example: fill mail queue directory
  ⇒ Set MaxMessageSize
  ⇒ BUT... setting this too low bounces legitimate mail
  ⇒ AND... attacker can send multiple messages
♦ Example: someone is mail bombing the machine
  ⇒ Set ConnectionRateThrottle
  ⇒ BUT... mail bomb still prevents legitimate mail from entering
Avoiding Denial of Service Attacks

♦ There is no magic bullet

♦ Try to balance previous options with these other options

⇒ DelayLA

• Delay new connections for one second instead of closing listening socket at specified load average

⇒ QueueLA

⇒ RefuseLA

⇒ Timeout.* options

⇒ MaxRecipientsPerMessage

⇒ BadRcptThrottle

• Limit the rate bad recipients (sleep for one second) are accepted via SMTP after reaching the specified threshold
Controlling Relaying and Other Access

♦ Spammers love open relays
♦ Theft of service
♦ Relaying denied by default \( (8.9) \)
  \[ \Rightarrow \text{May go overboard for some situations} \]
  • By definition, firewall and gateway configurations must relay from and to trusted hosts

♦ Relaxing anti-relay controls: \( (8.9) \)
  \[ \Rightarrow \text{FEATURE(`relay_entire_domain`)} \]
  \[ \Rightarrow \text{FEATURE(`access_db`)} \]
  • Reject or allow relaying by domain, IP address range, or e-mail address
  \[ \Rightarrow \text{Several other FEATURE()’s} \]
  \[ \Rightarrow \text{SMTP authentication} \ (8.10) \]
  • TRUST_AUTH_MECH()
  \[ \Rightarrow \text{STARTTLS certificates} \ (8.11) \]
  • Using CERTISSUER: and CERTSUBJECT: in access database
Privacy

♦ Meta-Information
  ⇒ Addresses
  ⇒ Delivery Status Notifications (DSN)

♦ Message privacy \(^{(8.11)}\)
  ⇒ SMTP AUTH encryption
  ⇒ STARTTLS encryption

♦ On disk
  ⇒ Mail queue
  ⇒ Mail store
    • Do not use set-group-ID local delivery agents and group readable or writable mailboxes if possible
  ⇒ Logs
    • Users should not be able to read mail logs
Privacy of Meta-Information

♦ A lot of information available via SMTP
  ⇒ E.g., spammers use VRFY and EXPN commands to collect addresses

♦ **PrivacyOptions=novrfy,noexpn** option can turn off VRFY and EXPN commands
  ⇒ Unfortunately, useful for debugging
  ⇒ *check_vrfy* and *check_expn* rulesets can control based on other info  

♦ **PrivacyOptions=noreceipts** turns off Delivery Status Notification (DSN) support for success return receipts
  ⇒ Disables DSN extension entirely  

♦ **PrivacyOptions=restrictexpand**  
  instructs sendmail to drop privileges when the \(-bv\) option is given
  ⇒ prevents users from reading private aliases, forwards, or `include:` files
  ⇒ Disables \(-v\) as well
Message Privacy

♦ Basically does not exist

♦ All mail is normally sent unencrypted
  ⇒ Some SMTP extensions change this
  ⇒ SMTP authentication with DIGEST-MD5 can also encrypt (but requires prior arrangements) (8.11)
  ⇒ STARTTLS extension will do opportunistic encryption (8.11)
  • Only guaranteed on systems under your control

♦ Suggest using end-to-end encryption
  ⇒ S/MIME, PGP, etc.
  ⇒ Requires Mail User Agent (MUA) support
  ⇒ Fits in well with verifying digital signatures
  ⇒ Only protects message body
  • Sender, recipients, headers still in the clear
SMTP Authentication (8.10)

- Based on Simple Authentication and Security Layer (SASL)
  ⇒ See RFC 2554 and RFC 2222

- Can be used to permit relaying based on shared secret

- SASL mechanisms
  ⇒ ANONYMOUS — No authentication
  ⇒ PLAIN — Password sent as clear text
  ⇒ CRAM-MD5 — APOP-style challenge/response system
  ⇒ DIGEST-MD5 — Stronger than CRAM-MD5
    • Supports optional security layer (8.11)
  ⇒ KERBEROS_V4
  ⇒ GSSAPI — Plugin framework (Kerberos v5)
  ⇒ EXTERNAL — Use external authentication information (e.g., from STARTTLS) (8.12)
  ⇒ others — SASL is extensible
Configuring SMTP Authentication (8.10)

♦ **AuthMechanisms**

⇒ List of mechanisms to advertise for incoming connections

⇒ Default: GSSAPI KERBEROS_V4
   DIGEST–MD5 CRAM–MD5
   • Also EXTERNAL (8.12)

♦ **TRUST_AUTH_MECH(‘mechanisms’)**

⇒ List of mechanisms that are trusted to allow relaying

♦ **DAEMON_OPTIONS(‘option-list’)**

⇒ New modifier flag M=a to require authentication for all connections

⇒ DAEMON_OPTIONS (‘M=a’)
   • Not legal on port 25 for machines advertised in MX records
   • See also “Controlling Server Features”
Client-Side SMTP Authentication  (8.10)

♦ Authorization identity (userid)
  ⇒ Identifier used to check whether operations are permitted

♦ Authentication identity (authid)
  ⇒ Identifier used to authenticate the client

♦ Secret
  ⇒ Password for authentication identity

♦ Realm
  ⇒ Specifies virtual “domain” for userid (optional)
  ⇒ Defaults to server’s hostname ($j)

♦ Mechanisms
  ⇒ List of mechanisms to try (defaults to AuthMechanisms)  (8.12)
Specifying Client Authentication

- **DefaultAuthInfo**  
  ⇒ File containing default information for outbound connections  
  ⇒ Recommend `/etc/mail/default-auth-info`  
  ⇒ Keep private (mode 0600)

- **authinfo** ruleset controlled by either:  
  ⇒ `FEATURE( access_db` ) or  
  ⇒ `FEATURE( authinfo` ) for separate database
  - Both use `AuthInfo: tag`  
    ```
    AuthInfo:domain       "U:userid"   "I:authid"
    "P:secret"   "R:realm"
    "M:mechanism"
    ```
    » Should be one line
  - Keep database private (mode 0600)
configuring Cyrus SASL

♦ .../lib/sasl/Sendmail.conf
♦ Plain text file containing lines with option: value
♦ pwcheck_method: how to check password
  ⇒ sasldb — read from a private database
  ⇒ passwd — read from /etc/passwd
  ⇒ shadow — read from /etc/shadow
  ⇒ PAM — use Pluggable Authentication Modules
  ⇒ kerberos_v4
  ⇒ pwcheck — use pwcheck daemon (supplied with Cyrus SASL)
♦ srvtab: where to find Kerberos V4 srvtab file
♦ auto_transition: if set to true, automatically add secrets to sasldb when PLAIN method used
Configuring Cyrus SASL (2)

♦ Realms
  ⇒ Groups of users
  ⇒ PLAIN, LOGIN, and CRAM-MD5 use user@realm hack for secrets look up
  ⇒ Other mechanisms have native support for realms

♦ Secrets database (sasldb)
  ⇒ Required for CRAM-MD5 and DIGEST-MD5
  ⇒ Update using saslpasswd
  ⇒ Must not be readable by users
Example SMTP AUTH Session

220 zim.gshapiro.net
ESMTP Sendmail 8.12.2

>>> EHLO gir.gshapiro.net
250-zim.gshapiro.net
  Hello pleased to meet you

250 AUTH DIGEST-MD5 CRAM-MD5
250 HELP

>>> AUTH DIGEST-MD5
334 PdgxNDA...Lm5ldD4=
>>> QG1vbmt...GE5YjcxZTJlMzI2YjY4N2M=
250 2.0.0 OK Authenticated

>>> MAIL From:<gshapiro@gshapiro.net>
AUTH=gshapiro@monkeyboy.gshapiro.net
250 2.1.0 <gshapiro@gshapiro.net>...
  Sender ok

>>> RCPT To:<msk@example.com>
250 2.1.0 <msk@example.com>...
  Recipient ok

⇒ Note relaying allowed due to successful authentication using a trusted mechanism
Transport Layer Security (TLS)  \((8.11)\)

- Based on RFC 2478
  - Encryption of SMTP connection; follow on to SSL
  - Uses X.509 certificates

- Buzzwords:
  - certificate — public part of key pair, includes identity information
  - key — private part of key pair
  - CA — certificate authority (signs certificates)
  - CI — certificate issuer (see CA)
  - DN — distinguished (unique) name, e.g.,
    
    
    \[/C=US/ST=California/L=Berkeley/O=Sendmail.ORG/CN=Sendmail\text{ CA}\]
  - CN — common name, e.g., host name or user’s full name (may not be unique)

- Must create and sign certificates for servers
- Note: STARTTLS is NOT end-to-end encryption
Digital Certificates

♦ Used to establish trust

♦ Hierarchical model — uses X.509 Certificate Authority (CA)
  ⇒ Trusted authority signs other certificates
  ⇒ Thawte, Equifax, Verisign, etc., or roll your own

♦ Server certificate
  ⇒ Certificate used for incoming connections
  ⇒ Identifies mail server to connecting client

♦ Client certificate
  ⇒ Certificate used for outgoing connections
  ⇒ Identifies connecting client to remote mail server
  ⇒ Often the same as server certificate
Configuring STARTTLS

♦ Setup certificates

⇒ Need both the signed certificate (public) and the certificate key (private, make sure permissions are safe)

⇒ Keys must not be encrypted (\texttt{--nodes})

♦ Client and server certificate Common Name (CN) should be the fully qualified domain name of the mail server

♦ Configure sendmail:

\begin{verbatim}
define(`CERT_DIR', `MAIL SETTINGS_DIR'`certs')
define(`confCACERT_PATH', `CERT_DIR/')
define(`confCACERT', `CERT_DIR/CAcert.pem')
define(`confSERVER_CERT',
  `CERT_DIR/SrvCert.pem')
define(`confSERVER_KEY',
  `CERT_DIR/SrvKey.pem')
define(`confCLIENT_CERT',
  `CERT_DIR/CltCert.pem')
define(`confCLIENT_KEY',
  `CERT_DIR/CltKey.pem')
\end{verbatim}
OpenSSL Certificate Creation

♦ Create Certificate Authority (CA) once

```bash
mkdir CA
cd CA
mkdir certs crl newcerts private
chmod 0700 private
echo "01" > serial
cp /dev/null index.txt
openssl req -new -x509 -keyout 
   private/cakey.pem -out cacert.pem
```

♦ Create certificate(s)

```bash
umask 0066
openssl req -nodes -new -x509 
   -keyout key.pem -out newcert.pem
```

♦ Sign new certificate(s) using CA

```bash
openssl x509 -x509toreq -in newcert.pem 
   -signkey key.pem -out csr.pem
openssl ca -policy policy_anything 
   -out cert.pem -infiles csr.pem
rm -f csr.pem
# (opt) rm -f newcert.pem (unsigned cert)
```
STARTTLS Operation \textit{(8.11)}

- **STARTTLS** should appear as an ESMTP extension in \texttt{EHLO} response; if not, check syslog for problem reports
  \[ \Rightarrow \text{Try: } \texttt{sendmail -bs -OLogLevel=14} \]

- **Received**: headers reflect **STARTTLS** usage

- **STARTTLS** can be configured via access database to:
  \[ \Rightarrow \text{Restrict incoming and outgoing connections} \]
  \[ \Rightarrow \text{Restrict recipient (RCPT) addresses based on TLS negotiation \textit{(8.12)}} \]
  \[ \Rightarrow \text{Allow relaying based on DN of certificate or DN of certificate issuer (CA) \textit{(8.11)}} \]
  \[ \Rightarrow \text{Control **STARTTLS** usage and negotiation \textit{(8.12)}} \]
Restrict Connections with STARTTLS(8.11)

♦ Looks up TLS_Srv: host-name-or-addr
  ⇒ Uses TLS_Clt: if acting as server
  ⇒ Subdomains and subnets also tried

♦ If not found, looks up TLS_Srv: or
  TLS_Clt: for default behavior

♦ If not found, allows connection (but not relaying)

♦ Value (RHS) must be one of
  ⇒ VERIFY — certificate verification required
  ⇒ ENCR: bits — must be encrypted with at least bits encryption strength
  ⇒ VERIFY: bits — both

♦ If found in database and conditions do not match, reject connection
  ⇒ RHS can have TEMP+ or PERM+ to denote temporary or permanent rejection
    • Default is temporary unless
      TLS_PERM_ERR is set via m4
Restrict Connections with STARTTLS(8.11)

♦ Can also have a list of extensions, the first extension starts with + and additional extensions are separated by ++  \hspace{1cm} (8.12)

⇒ CN: name — CN must be name
⇒ CN — CN must be name of server
⇒ CS: name — DN must be name
⇒ CI: name — CI DN must be name

♦ Examples
⇒ Incoming from 10.213.23.10: require verified certificate and \geq 112 bit encryption
\begin{verbatim}
TLS_Clt:10.213.23.10 VERIFY:112
\end{verbatim}
⇒ Outgoing to *.sendmail.org: require verified certificate and \geq 112 bit encryption
\begin{verbatim}
TLS_Srv:sendmail.org TEMP+VERIFY:112
\end{verbatim}
⇒ Outgoing to smtp.sendmail.com: require \geq 112 bit encryption and CN of smtp.sendmail.com else permanent failure
\begin{verbatim}
TLS_Srv:smtp.sendmail.com PERM+ENCR:112+CN:smtp.sendmail.com
\end{verbatim}
Restricting Recipients for STARTTLS (8.12)

♦ Looks up \texttt{TLS\_Rcpt:user@domain}, \texttt{TLS\_Rcpt:user@}, \texttt{TLS\_Rcpt:domain}, \texttt{TLS\_Rcpt:}:

⇒ Tries subdomains after \texttt{TLS\_Rcpt:domain}
⇒ First match wins
⇒ Same values (RHS) as \texttt{TLS\_Srv/TLS\_Clt}

♦ Useful for messages with multiple recipients to different domains

♦ Example

⇒ If delivering to gshapiro@eng.sendmail.com recipient, must be talking to a Sendmail, Inc. CA

\texttt{TLS\_Rcpt:sendmail.com} \quad \texttt{TEMP+VERIFY:112+}
\texttt{CI:/C=US/ST=California/}
\texttt{L=Emeryville/O=Sendmail,+20Inc/}
\texttt{OU=IT/CN=Sendmail+20Cert/}
\texttt{Email=rootca@sendmail.com}

• Note certificate value folding for readability
Allowing Relaying with STARTTLS (8.11)

♦ Look up certificate issuer in access database using CERTISSUER: tag with values:
  ⇒ RELAY — allow relaying
  ⇒ SUBJECT — look up certificate subject using CERTSUBJECT: tag

♦ Examples
  ⇒ Allow gshapiro.net CA signed certificates to relay

CertIssuer:/C=US/ST=California/L=Emeryville/
  O=gshapiro.net/CN=CA/
  Email=certs@gshapiro.net      RELAY

  ⇒ If Sendmail, Inc. CA signed certificate...

CertIssuer:/C=US/ST=California/L=Emeryville/
  O=Sendmail,+20Inc./CN=Sendmail+20Cert/
  Email=rootca@sendmail.com      SUBJECT

  ⇒ ... and that certificate belongs to gshapiro@sendmail.com, allow it to relay

CertSubject:/C=US/ST=California/L=Emeryville/
  O=Sendmail,+20Inc./OU=Engineering/
  CN=Gregory+20Neil+20Shapiro/
  Email=gshapiro@sendmail.com      RELAY

• Note certificate key folding for readability
Controlling STARTTLS Client  \( (8.12) \)

- By default, STARTTLS is requested on all outgoing connections and offered on incoming connections when certificates are configured.

- Can turn off outgoing STARTTLS using access database `Try_TLS` tag with RHS of **NO**

  \[ \Rightarrow \text{Can use hostnames or IP addresses after tag} \]

  \[ \Rightarrow \text{Subdomains and IP networks checked} \]

  \[ \Rightarrow \text{Examples} \]

  \[ \Rightarrow \text{Turn off outgoing STARTTLS to} \]

  \[ *\text{.example.com, example.com and 127.0.*; however, offer it for tls.example.com} \]

```
Try_TLS:tls.example.com  YES
Try_TLS:example.com     NO
Try_TLS:127.0           NO
```
By default, SMTP AUTH and STARTTLS are offered for incoming connections and a client certificate is requested for STARTTLS

⇒ Can change using Srv_Features: tag
⇒ Same left hand side usage as Try_TLS:
⇒ Value is one or more of the following space separated characters:
  • S —Do not offer STARTTLS
  • V —Do not request STARTTLS client cert
  • A —Do not offer SMTP AUTH
     » Lower case means opposite (s means offer STARTTLS)

⇒ Examples
  • Change default to not offer SMTP AUTH
  • Do not offer STARTTLS to smtp.example.com but offer SMTP AUTH

Srv_Features: A
Srv_Features:smtp.example.com S a
Logging

♦ **LogLevel** option controls how much logging is done

♦ Usual default (**LogLevel**=9)
  ⇒ Message receipts, deliveries, failures
  ⇒ Exceptional conditions

♦ Increasing yields more information
  ⇒ Alias expansions
  ⇒ Incoming SMTP protocol
  ⇒ Much more and varied information available

♦ Logs using *syslog*(3), **mail** facility

♦ Little point in logging if logs are not checked
Dealing with Forgeries

♦ Problem: someone forges mail from your domain

♦ Bad news: can not do anything if mail originates from afar
  ⇒ Common spammer profile
  ⇒ Might be able to gather information from Received: headers

♦ Good news: can do quite a bit if mail comes from a machine you control
  ⇒ IDENT information in Received: headers from trusted machines

♦ Generally, do not assume a sender is who they say they are without other validation, e.g., digital signature

♦ PrivacyOptions=authwarnings can expose a lot of attempts at local forgery
Tracing Forgeries

♦ **Received:** headers show:

⇒ Hostname, IP address, and possibly IDENT of connecting server

from knecht.Neophilic.COM (knecht.sendmail.org [209.31.233.176])

⇒ Hostname, possibly version, and protocol used on receiving server

by horsey.gshapiro.net (8.12.2/8.12.2) with ESMTP id f717AT042;

⇒ Transport characteristics such as SMTP AUTH and STARTTLS information if negotiated (8.11)

(version=TLSv1/SSLv3 cipher=EDH-RSA-DES-CBC3-SHA
bits=168 verify=OK)

⇒ Date of transaction

Wed, 1 Aug 2001 00:10:31 -0700 (PDT)

♦ Ordered from most recent to least recent

♦ Should only trust the **Received:** headers added by trusted machines
IDENT Server

♦ If client sends mail to server, server can do an IDENT query back to client asking who is sending the mail
   ⇒ Client need not respond to IDENT query
   ⇒ Many firewalls block IDENT port (TCP 113)

♦ Result can be plain text or encrypted
   ⇒ Opaque as far as server is concerned
   ⇒ Warning: client might lie

♦ Server puts that info in Received: header
System Taxonomy

- Systems with user login access
- Systems with user accounts, but not logins (mail hub)
- Systems with POP/IMAP access only (mail servers)
- Systems with SMTP-only access (firewalls)
Systems with User Logins

♦ Things need to be in the “right place”
  ⇒ User programs may expect them there

♦ sendmail daemon needs to run as root
  ⇒ Assume user identity to read or write files and execute programs
  ⇒ May only need to run sendmail daemon on a mail hub
    • Still need periodic queue runs in case mail hub was unavailable

♦ Need a set-group-ID sendmail binary for mail submission
  ⇒ Set-user-ID root prior to 8.12

♦ Can still have tight security policy
  ⇒ But there are limits
Securing Systems with User Logins

- `sendmail.cf` (and directory path) writable only by `root` or `TrustedUser`  
  ⇒ Same for `submit.cf` (8.12)

- Maps writable only by `root` or `TrustedUser`
- Map directory paths secured (`root` or `TrustedUser`)
- Queue directories secured
- Use `smrsh` to control user program selection
- Set `SafeFileEnvironment` to control file writing
- Aliases file clean
- Consider avoiding forward files
  ⇒ Create a forward and `vacation` UI that updates an aliases file
  • Do not allow user program or file delivery since aliases run as `DefaultUser`

  ⇒ Use `procmail` as LDA so users can forward or deliver to files and programs
  • But makes `smrsh` useless
Systems with User Accounts, No Shell Access

♦ Forward files still work
  ⇒ Presumably accessed via NFS
  ⇒ Might use some batch update mechanism
  ⇒ Consider previous suggestions

♦ May mean that user programs can be run
  ⇒ Controlled by /etc/shells file
    • Include /SENDMAIL/ANY/SHELL/ to allow any user to run programs
  ⇒ smrsh can control choice

♦ May be able to turn off set-user-ID (pre-8.12) or set-group-ID (8.12 and later) bit on sendmail binary
  ⇒ Only if no programs or scripts call /usr/sbin/sendmail directly
  ⇒ However, many do (cron, Majordomo, vacation, CGI scripts, ...)

♦ Otherwise much like user shell access case
  ⇒ May want to avoid procmail suggestion
Securing Systems with User Accounts, No Shell Access

♦ All previous comments apply
♦ \textbf{SafeFileEnvironment=/home} to limit file write attacks to home directories
♦ Can disable delivery to files if not used
  \Rightarrow Remove \texttt{F=/} from local mailer definition
  \Rightarrow Redefine \texttt{*file*} mailer
    \quad \texttt{M*file*, P=/bin/false}
♦ Consider moving forward files out of home directory
  \Rightarrow \texttt{ForwardPath=/var/forward/$u}
  \Rightarrow Avoids NFS glitches
♦ Can disable delivery to programs if not used
  \Rightarrow Remove \texttt{F=|} from local mailer definition
  \Rightarrow Redefine \texttt{*prog*} mailer
♦ May be able to disable :include: syntax
  \Rightarrow Remove \texttt{F=:} from local mailer definition
POP/IMAP Access Only (Mail Server)

♦ Generally, no user access

♦ May still have user accounts
  ⇒ Possible to modify sendmail and POP/IMAP server to use alternate user database
    • Cyrus IMAP (includes POP server) already does this

  ⇒ Considerably more secure

♦ Does not use forward files
  ⇒ Well, maybe if there is a web agent to update them

♦ sendmail need not run as root (after startup)
  ⇒ No user programs will access sendmail directly
  ⇒ Except, of course, for cron, Majordomo, etc.

♦ May be other issues with the POP/IMAP server
Securing Mail Servers

♦ Set `RunAsUser` option to have sendmail cede `root` privileges as soon as possible
♦ May be able to turn off set-user-ID (pre-8.12) or set-group-ID (8.12 and later) bit on sendmail binary
  ⇒ As long as not directly invoked by user processes, e.g., `cron`, Majordomo, etc.
♦ Easier to run in `chroot` or `jail` area
Running with Majordomo

♦ The sendmail binary must retain set-user-ID root (pre-8.12) or set-group-ID smmsp (8.12 and later) privileges

♦ Majordomo defaults to group writable directory for :include: files

⇒ sendmail 8.9 and later do not like that
⇒ Not necessary for proper operation
⇒ Can be fixed by tweaking the permissions as described in sendmail FAQ 3.32:

   http://www.sendmail.org/faq/section3.html#3.32

♦ Put Majordomo aliases file in /etc/mail/ instead of in Majordomo’s directory
Firewall Configuration

♦ No user accounts or logins
♦ No per-user forwarding
♦ No local mail delivery
  ⇒ SMTP in and out only
♦ Some routing, mostly simple
♦ Aliasing acceptable
  ⇒ Only to other addresses, not to files or programs
♦ Possible external services, e.g., Majordomo, but strongly discouraged
Securing Firewall Configurations

♦ Set RunAsUser
  ⇒ Truly paranoid users have been known to eliminate the need to even start sendmail as root
    • Listen on high port (e.g., 2525) instead of 25
    • Use firewall (ipfw, ipf) software to redirect port 25 traffic to high port

♦ Set SafeFileEnvironment

♦ Disable prog mailer if not needed (should not be)

♦ Disable *file* mailer if not needed (should not be)

♦ Run in a chroot or jail area

♦ In summary, firewall should just relay messages, not act upon them
Parting Advice

♦ Keep up to date

⇒ Subscribe to sendmail-announce
sendmail-announce-request@lists.sendmail.org

⇒ Need to pay attention to more than just sendmail updates
  • Operating system updates
  • Berkeley DB, BIND, Cyrus SASL, OpenLDAP, OpenSSL, etc.

⇒ Subscribe to security mailing lists
  • BugTraq
  • CERT advisories

♦ Double check permissions, aliases

⇒ Make sure procmail is not in smrsh directory

♦ Create mailnull account (8.9)
Parting Advice

♦ Consider blocking incoming port 25 at the site firewall for all machines except mail servers
⇒ Relay all incoming and outgoing mail through those few secured firewall (SMTP only) machines
  • Easier to secure and monitor
  • Easier to implement anti-spam and anti-relaying policy
⇒ Block port 587 completely from the outside

♦ Actually check the logs on a regular basis
For More Information

♦ Tutorials
  ⇒ Eric Allman’s *Sendmail Configuration and Operation* Tutorial

♦ Read The Fine Manuals (RTFM)
  ⇒ *cf/README*
  ⇒ *doc/op/op.ps*
  ⇒ *sendmail/README*
  ⇒ *sendmail/SECURITY* (8.12)
  ⇒ O’Reilly’s *sendmail* book, 2nd Edition

♦ On the Internet
  ⇒ sendmail FAQ: http://www.sendmail.org/faq/
  ⇒ Sendmail Consortium:
    http://www.sendmail.org/
  ⇒ Sendmail, Inc: http://www.sendmail.com/
  ⇒ USENET: comp.mail.sendmail
  ⇒ Questions: <sendmail@sendmail.org>