Designing and Securing Modern TCP/IP Networks

Tina Darmohray
Lawrence Livermore National Laboratories
tmd@s1.gov

Rob Kolstad, Ph.D.
BSDI, Inc.
kolstad@bsdi.com

Contributions from Trent Hein, Evi Nemeth, Lee Neely

Schedule

Schedule

Today's Topics

- Today's Schedule
- Networking problem definition
- Configuring mailers
- Domain Name System
- Firewalls: Routers
- Firewalls: Gateway hosts
- Firewalls: "Proxy" users
- Firewalls: DNS
- Firewalls: Mail
- Ethics in systems administration and at your site

Problem Definition

Networking

- "Networking solves all problems"
- Networking's evolution became a revolution
  - Scaling failure surprised many
  - Security problems surprised some
  - Administration problems are astounding

What Networking Does

- Networking enables computers to exchange data
  - File transfer & filesystem sharing
  - Remote login
  - Electronic mail
  - X terminals
  - Low level directives (e.g., remote execution)
  - Time synchronization
  - Support services (e.g., naming and packet routing)

Kinds of Networking

- Small sites use "Local Area Networking" for geographically close computers – often in "Workgroup Computing"
- "Wide Area Networking" covers greater distances (usually more slowly) – "Enterprise-wise Computing"
- Homogeneous – same computer brand
- Heterogeneous – different kinds of computers (e.g., Sun, IBM PC, & Mac)

How Hard Can It Be?

- Constructing networks is challenging
  - Hardware installation
  - Software configuration
- Must plan ahead
  - Complete functionality
  - Scalability
  - Security

Foundations of Network Construction

- Understand networking functionality
- Understand physical installation and its debugging
- Understand configuration and administration
  - Debugging
  - Maintenance
  - Scaling
- Highlight routing and naming (and their debugging)
- Review security problems and offer solutions
- Understand monitoring and performance issues

How We'll Proceed

- We'll proceed through the blocks necessary to construct the solution including:
  - Long section on Domain Naming
  - Long section on Security
  - Long section on Firewalls

Sendmail

Outline

- Implementation of firewalls requires re-writing addresses so that internal machine names are not passed outside the corporate net
- `sendmail` is the program that rewrites addresses
- We'll spend two hours taking apart sendmail.cf files so that you can make it do almost anything you want

`sendmail`

- The most common internetwork mail router in use is the `sendmail` program written by Eric Allman
- Collects messages generated by user interface programs,
- Reformats the headers as required
- Rewrites the addresses as necessary
- Chooses and calls appropriate mailers for message delivery

Mail System Functions

- The mail system has three major functions
  - Interface
  - Routing
  - Delivery

Interfaces

- The mail system includes the user interfaces to collect and display messages
  - **/bin/mail**
  - **/usr/ucb/mail**
  - **/usr/new/mh/{inc,show,forw,...}**
  - **elm**
  - **pine**
  - mush
  - Even **emacs**

Routing

- The mail system includes the routing mechanism to decide how to deliver messages
  - **sendmail** – the real workhorse with its configuration file **sendmail.cf**
  - **MMDF** (**PMDF** in Pascal)
  - **SMAIL** (3.1.21 in alpha test 6/91)

Delivery

- "Delivery" is the depositing of the mail text into a user's mailbox or transporting mail from one site to the next

- Mailers to do actual delivery
  - Local – within this machine
  - **UUCP**
  - **CSNET** – **mmdf** or **pmdf**
  - **BITNET** – jnet (VMS) & urep (UNIX)
  - Internet – **SMTP** over **TCP/IP** (integrated into **sendmail**)

What sendmail Does

- **sendmail** has a main chore that is the focus of our studies:

  sendmail transforms an address into a (username, remote_machine, mailer_prog) triple

- **sendmail** also has some other functionality:
  - It rewrites sender and receiver addresses if necessary (e.g., to hide internal machine names or add domain info)
  - It controls the headers put on outgoing mail messages (never touch this – consider them standard as in RFC 822)
  - It delivers messages using TCP/IP (we'll not discuss the protocol)
  - It receives and delivers incoming messages (we'll not discuss this)

The Configuration File

- The configuration file for **`sendmail`** is called **`sendmail.cf`** and lives in either **`/usr/lib`** or **`/etc`**
- It contains:
  - Comments (lines starting with '#' are ignored)
  - Blank lines (also ignored)
  - Definitions of variables (one letter names)
  - Rewriting rules (designated S0, S2, and so on)
  - Miscellaneous other lines that parameterize sendmail
  - Mailer definitions
- It is difficult to read for the neophyte; easy for you after this presentation

An Input-Output Model for `sendmail`

- **`sendmail`** processes three addresses for mail to a single recipient:
  - Destination address → (user, machine, mailer)
    `kolstad@sun.com → (kolstad, sun, ether)`
  - Sender address → Rewritten sender address
    `evi → evi@boulder.Colorado.EDU`
    or
    `evi → boulder!evi`
  - Recipient address → Rewritten recipient address
    `ace!ralph → ralph@ace.uucp`

The First Mystery

- Unlike many programming languages, **`sendmail`** does not execute its configuration file from top to bottom
- First, it determines the triple of
  (user, remote_machine, mailer)
  - It applies ruleset 3 to recipient's address
  - It applies ruleset 0 to the result
  - Note: ruleset 0 must 'resolve' to the triple
- This is a bit like a program which creates an envelope for the message

More of The First Mystery

- Second, it reformats the sender's mail address in the header (may be a no-op):
  - It applies ruleset 3 to sender's address
  - It applies ruleset 1 to the result
  - It applies the rule specified as 'S=' in the mailer resolved above (if the 'S=' field is present) (e.g., uucp rewriting)
  - It applies ruleset 4 to the result
  - Result of these rules is the final sender address in the headers

The Rest of The First Mystery

- Third, it reformats the recipient's mail address in the header (may be a no-op):
  - It applies ruleset 3 to recipient's address
  - It applies ruleset 2 to the result
  - It applies the rule specified as 'R=' in the mailer resolved earlier (if specified)
  - It applies ruleset 4 to the result
  - Result of these rules is final recipient address in the headers

Summary of Rule Evaluation Order

- Address resolution: 3, 0
- Sender: 3, 1, mailer-specific, 4
- Recipient: 3, 2, mailer-specific, 4
- The mailer-specific rules are specified as `S=` and `R=` in the `M` mailer definition lines

Simple Examples

- Consider user 'kolstad' sending a UUCP mail message to user 'beast!dave'
- Ruleset 3 on our system transforms 'beast!dave' into 'dave<@beast.uucp>', a typical canonical format for an address
- Ruleset 0 on our system resolves that address into
  (beast!dave, uunet.uu.net, ether)
  which punts the address to uunet, which is smarter than we are
- Local messages require fewer machinations: `mail polk` resolves to (polk, , local)

The Configuration File

- Lots of documentation refers to certain, fixed configuration files with definite uses for variables and rules
  - Only the rules mentioned previously are nondynamic
  - Lower-case variables have meanings fixed by `sendmail` (a table is included later)
  - Upper-case variables are user-definable – meanings are fixed by the configuration file, not `sendmail`
- We'll talk mostly about modifying existing configuration files but will not concentrate on any particular vendor's or site's files
- Regard vendor files' comments with skepticism

General Format of Commands

- Remember FORTRAN with its fixed-column input? This is that arcane...
- Many commands have this format:

```
XVstuff
↑↑  ↑
||  info to complete command
|variable that command affects
which command
```

Here's an example:
DAzzz
which defines variable A to have the string value **zzz**

General Format, continued

- Some commands have only:

```
Xstuff
↑↑
|info to complete command
which command
```

Here's an example:
S23
which means start ruleset 23

- Rewrite rules are discussed later

Variables

- Variables come in two flavors with separate namespaces: macros (strings scalars) and classes (associative arrays)
- Names of variables are single letters
  - **sendmail** owns the lowercase variables
  - Users own the uppercase variables
- Macros
  - Single valued (one string)
  - Defined like this:

```
DUsmith
DDsun.com ($v)
```

- Interpolate (dereference) variables using the dollar sign (**sendmail** binds very early):
  ... $U ... → **... smith ...**

Class Variables

- Classes
  - Multiple valued (potentially many words)
  - Defined like this:

```
CWsummit lucifer xyzzy
FX/usr/etc/workstation.names
FZ/usr/local/people %*s:%s
```

  - The **C** appends the three words listed to class **W**
  - The **F** directive specifies appending words in the named file (one per line) to class **X**; it can process an optional **sscanf** format
- Used in different contexts than simple expansion (e.g., **$=X** asks if a token matches any in class X)

Progress So Far

- Our hub configuration file has 392 lines
- 100 of them (about 25%) are blank
- 131 of them (about 33%) are # comments
- 6 of them (1.5%) are class definitions (machine names for hiding, mostly)
- 10 of them (2.6%) are variable definitions (mostly mail header definitions)
- You can now read 100+131+6+10=247 (63%) of the configuration file!

Header Definitions

- Mostly, don't mess with these; they conform to standards which other mailers expect
- Here's the most complex one:

```
HReceived: $?sfrom $s $.by $j ($v/$V)
        id $i; $b
```

- Note the continuation line (a tab marks it)
- Conditional macro substitution is a bit bizarre:

```
    HReceived: $?sfrom $s $.by $j ($v/$V)
Conditional:   ↑↑---------↑↑
```

Means: if **s** is defined, include the phrase "**from $s**"

Example of Expanding a Header

- Let's define some macros for the 'Received' line:
  - The **s** macro is pre-defined as the sender's hostname (say, **summit**)
  - The **j** macro, though not pre-defined, is the fully qualified hostname (say, **rmtc.Sun.COM**)
  - The **v** macro is the pre-defined sendmail version number (say, **4.1**)
  - The **V** macro is the local version number (say, **4.7**)
  - The **i** macro is the queue id of the message (say, **AA10874**)
  - The **b** macro is the current date (say, **Sun, 30 Sep 90 08:11:21 MDT**)

  All pre-defined macros are listed at the end of this section

Example of Expanding a Header, cont.

- So, the header lines:

```
HReceived: $?sfrom $s $.by $j ($v/$V)
    id $i; $b
```

expand to:

```
Received: from summit by rmtc.Sun.COM (4.1/4.7)
        id AA10874; Sun, 30 Sep 90 08:11:21 MDT
```

- Note that a little definition in the header line went a long way

The Header Definition Set

- Here's the complete standard header definitions:
```
H?P?Return-Path: <$g>
HReceived: $?sfrom $s $.by $j ($v/$V)
        id $i; $b
H?D?Resent-Date: $a
H?D?Date: $a
H?F?Resent-From: $q
H?F?From: $q
H?x?Full-Name: $x
HSubject:
H?M?Resent-Message-Id: <$t.$i@$j>
H?M?Message-Id: <$t.$i@$j>
```


About Those Headers

- The syntax is easy except for the '?'s which mean 'if this flag is specified for this mailer (on its M line's F specification), include this output'
- Headers from input are propagated regardless of the specs on the previous slide

Setting Options

- Option setting is easy once one realizes that most **sendmail** commands start with a few characters that must be read one at a time:
```
OQ/usr/spool/mailq
|||  ---value----  |
|Q -- which option to set
O -- set an option
```
- The same options may be set using the **-o** flag on the invocation line
- The *Sendmail Installation and Operation Guide* (SIOG) lists in Appendix B over 30 options you can set
- They really aren't worth covering here; many are obscure

Precedence Fields

- Define numerical precedence values like this standard set:
```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```
- Other than the fact that lower numbered precedence mail have the property that errors are not returned for them, these numbers are not yet processed by an extant mailers (an extension that hasn't made it yet)

Trusted Users

- Users who are permitted to override the send address using the `-f` flag to `sendmail` are listed as trusted users.
- Be careful!
- Here's a (the?) typical list:

```
Troot
Tdaemon
Tuucp
```

- Be sure the wizard password is known to no one (the `OW` "oh-W" option used to set it)

Mailer Definitions

- Mailer definitions specify the mailer program to call for delivery along with:
  - Options to use (e.g., with `H` specifications)
  - Parameters to use
  - Mailer specific rewriting rules for the sender
  - Mailer specific rewriting rules for the recipient
- Here's a typical set of mailer definitions:

```
Mlocal, P=/bin/mail, F=rlsDFmn, S=10, R=20,
    A=mail -d $u
Mprog,  P=/bin/sh, F=lsDFe, S=10, R=20, A=sh -c $u
Mether, P=[IPC], F=msDFMueCX, S=11, R=21, A=IPC $h
Muucp,  P=/usr/bin/uux, F=sDFhuU, S=13, R=23,
    M=120000, A=uux -p -a$g -z -gC $h!rmail ($u)
Mpathuucp, P=/usr/lib/pathuux, F=sDFhuU, S=13,
    R=23, M=120000, A=pathuux $h $u
```


Mailer Definition Anatomy

- Consider:

```
Muucp,  P=/usr/bin/uux, F=sDFhuU, S=13, R=23,
    M=120000, A=uux -p -a$g -z -gC $h!rmail ($u)
```

- The `M` specification tells us that this mailer's name (for resolution purposes) is `uucp`
- The `P` flag says the path of the program to execute to deliver this message is `/usr/bin/uux`

(continued)

More Mailer Definition Anatomy

- Still examining:

```
Muucp,  P=/usr/bin/uux, F=sDFhuU, S=13, R=23,
    M=120000, A=uux -p -a$g -z -gC $h!rmail ($u)
```

- The `F` field contains the flags to be set
  - conditionally interrogated for headers
  - Mailer flags (23 of them) shown in Appendix C of *SIOG*
- The `S` and `R` fields specify the name rewriting rules for the sender and recipient fields

More Mailer Definition Anatomy

- **Still examining:**
```
Muucp,  P=/usr/bin/uux, F=sDFhuU, S=13, R=23,
    M=120000, A=uux -p -a$g -z -gC $h!rmail ($u)
```
- **The `M` field tells the maximum size message for this delivery mechanism; 50,000 usually isn't enough; no value for `m` means infinity; this is NOT end-to-end limit (System V `ulimit` still counts)**
- **The `A` field contains the argument vector for exec'ing `/usr/bin/uux`; note use of macros**

Progress So Far

- **We've added:**
  - 10 header definitions
  - 3 precedence definitions
  - 3 trusted users
  - 19 configuration lines
  - 5 mailer definitions
- **Now we can confidently read 247+10+3+3+19+5=287 (73%) of the configuration file**
- **Only about a quarter of the total lines are rules and rule headers, but they're the most important (and the trickiest)**
- **On to the good stuff**

Rewriting Rules

- **Unix mail addresses can be fairly complex:**
  - `user`
  - `user@site`
  - `vaxsite::user`
  - `user@rmtc.subd1.subd2.domain`
  - `site!user`
  - `user@site.dom1@site.dom2` (may be illegal)
  - `uucpsite!user@site1.dom`
  - `site!user@uunet.uu.net`
  - `user%host@site.dom`
- **Sometimes complex addresses nevertheless represent local mail**
- **The rewriting rules figure out where each message must go next and what address that site should see and how it should get there (the famed triple from earlier slides)**

Execution of Rewrite Rules

- **The key to success in `sendmail` is understanding the behavior of rewrite rules**
- **The rules operate on a given address**
- **These rules specify a pattern and a replacement string (semantically like an editor substitution command)**
- **If the pattern matches (details later), it is replaced by the replacement string – and the statement is re-executed (usually)**
- **The statement continues until the match fails**

Rewriting Rule Syntax

- Rewriting rules look like this:

```
R1hs    rhs      comments
↑ ↑      ↑           ↑
| |      |       optional
| |   the new pattern
| search pattern for what gets changed
R -- rule specifier
```

- NOTE: TABS must separate lhs, rhs, and comments (this means lhs and rhs can contain blanks)
- Comments written by others may be misleading

Execution

- lhs must match address completely and exactly for substitution to occur (except upper/lower case distinctions are ignored)
- Matching uses tokens
  - Define delimiter characters in pre-defined variable `o`:
    `Do.!:@^%/[]`
  - Tokens are delimiters or those things which come before, after, or between delimiters
  - A statement re-executes as long as the match works (look out for infinite loops!)
  - Note that matching 'pessimizes'
- Example: `user%host@site.dom` has seven tokens:
  `user % host @ site . dom`

LHS Pattern Matching

- Special tokens

	token
$*	Match 0 or more tokens
$+	Match 1 or more tokens
$–	Match exactly 1 token
$=X	Match any token in class X
$~X	Match any token NOT in class X

- Easy example showing `$` mechanism of substitution:

```
R$+@summit.central.sun.com  ↑ $1@rmtc  ↑ hide summit
R        (lhs)                (rhs)      (comment)
                        [TAB]      [TAB]
```

so `kolstad@summit.Central.Sun.COM` becomes `kolstad@rmtc` (`kolstad` is `$1`)

- Matching pessimizes

RHS Substitution Specification

- The rhs contains text and/or special symbols for substitution; here are some of the special symbols:

RHS token	Meaning
`$n`	Use wildcard match `n` from the LHS
`$>narg`	Call ruleset `n` with `arg` as input
`$:`	Prefix → 'apply rule only once'
`$@`	Prefix → 'return RHS as ruleset result now'
`$#mailer`	Resolve mailer to `mailer` (S0 only)
`$@host`	Resolve host to `host` (S0 only)
`$:user`	Resolve user to `user` (S0 only)

- Consider `R$+    $:$>7$1`
  - Matches everything
  - Passes everything (`$1`) to ruleset 7
  - No infinite loop because of `$:`

Examples of Rewriting Rules

- Add the domain name to the local address
  - Domain here (`$D`) is "`Colo.EDU`"
  - Mail machine (`$w`) is "`bou`"
  - For now, have omitted < and >

```
    R$+@$w               $1@$w.$D
    evi@bou      →     evi@bou.Colo.EDU
 evi%dw@bou      →  evi%dw@bou.Colo.EDU
   eric@ucbvax   →    eric@ucbvax    (No match)
 vaxa::kjk       →    vaxa::kjk      (No match)
```

- Handle the `::` for DECNET

```
 R$-::$-           $2@$1.DECNET
vaxa::kjk      →  kjk@vaxa.DECNET
spot::evi@bou  →  spot::evi@bou    (No match)
 evi@bou       →  evi@bou          (No match)
```


Greedy Matching

- Note how greedy matching works:

```
      R$+@$+              $1@$w.$D
      evi@bou       →     evi@bou.Colo.EDU
  evi@bou@bou       →     evi@bou@bou.Colo.EDU
```

- Every once in a while, the greedy matching surprises you

More Examples of Rewriting Rules

- On master mail machine, hide the names of individual local machines behind the name of the master mail machine
  - Class A (hidden workstations) contains host 'anchor'
  - Class A does NOT contain host 'vaxf'

```
R$+<@$=A>           $@$1<@$w.$D>
evi<@anchor>   →  evi<@bou.Colo.EDU>
evi<@vaxf>     →  evi<@vaxf>        (No match)
blia!eric      →  blia!eric         (No match)
```

- Those brackets (`<...>`) are a standard technique for focusing the real host of interest
- The `$@` token means "return this value" (as from a function)

A Ruleset

- Consider Ruleset 0, which (ultimately) will resolve the address:
  - The `S` specification introduces a ruleset
  - This ruleset has more rules coming on future slides

```
S0
# R3 has focused the <>
# delete the local inf
#  $w is the current hostname
R$*<$*$w.LOCAL>$* $1<$2>$3 thishost.LOCAL
R$*<$*$w.UUCP>$*  $1<$2>$3 thishost.UUCP
R$*<$*$w>$*       $1<$2>$3 thishost
R$*<$*.>$*        $1<$2>$3 drop trailing dot
R<@>:$*           $@$>0$1  retry after route strip
R$*<@>            $@$>0$1  strip null trash & retry
# Eng mail to exodus:
R$*<@Eng>       $#ether$@exodus.Eng.Sun.COM$:$1@Eng
```

More Ruleset S0

(earlier:)

```
CN ARPA EDU MIL GOV COM ORG
CN NET csnet bitnet
# if it goes to Sun.COM, send it to snail
R$*<@$*Sun.COM> $#ether$@snail.Corp.Sun.COM$:$1@$2Sun.COM
# sun-barr is smarter about domains, ship 'em off
R$*<@$*.$=N>   $#ether$@sun-barr.Sun.COM$:$1@$2.$3
# so far, we've serviced everything but local, uucp, and unknown
# resolve local hosts we hide that mount our spool/mail dir
R$*<@$=Y>$*        $#local$:$1   host rmtc hides
R$*<@$=Y.UUCP>$*  $#local$:$1   ..with .UUCP
# resolve UUCP names that can go direct
R$+<@$=D.$+.UUCP>       $#uucp$@$2$:$3!$1
R$+<@$=D.UUCP>          $#uucp$@$2$:$1      user@host.UUCP
```


More Ruleset S0

```
# route straight to uunet via sun-barr
R$+<@$-.$+.UUCP> $#ether$@sun-barr.Sun.COM$:$2.$3!$1@uunet.UU.NET
R$+<@$-.UUCP>    $#ether$@sun-barr.Sun.COM$:$2!$1@uunet.UU.NET
# what if it's a domain and we don't know? Send it to snail
R$*<@$+.$->      $#ether$@snail.Corp.Sun.COM$:$1@$2.$3
# remaining names must be local
R$+              $#local$:$1      everything else
```

- **Once you figure out how to read the rules and how the substitution mechanism works, you're home free**
- **Don't reference uunet.uu.net explicitly unless your mailer does MX records correctly – uunet forwards its mail a lot**

Testing Rewriting Rules the Hard Way

- **Run `sendmail` in address test mode; specify which ruleset to run after ruleset 3 and the address to be tested (note that ruleset 3 runs for free):**

```
% /usr/lib/sendmail -bt -Csendmail.cf.new
ADDRESS TEST MODE
Enter <ruleset> <address>
> 0 kolstad@usenix.org
rule 3   input: "kolstad" "@" "usenix" "." "org"
rule 8   input: "kolstad" "@" "usenix" "." "org"
rule 8 returns: "kolstad" "@" "usenix" "." "org"
rule 6   input: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 6 returns: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 3 returns: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 0   input: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 3   input: "kolstad" "@" "usenix" "." "org"
rule 8   input: "kolstad" "@" "usenix" "." "org"
rule 8 returns: "kolstad" "@" "usenix" "." "org"
rule 6   input: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 6 returns: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 3 returns: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 0 returns: $# "ether" $@ "sun-barr" "." "Sun" "." "COM"
                   $: "kolstad" "@" "usenix" "." "ORG"
```


Testing Sender Rewriting Rules the Hard Way

- **Note that one must be cleverer when doing sender rewrite rules:**

```
% /usr/lib/sendmail -bt -CNewConfFile.cf
> 2,13,4 rob
rule  3   input: "rob"
rule  3 returns: "rob"
rule  2   input: "rob"
rule  2 returns: "rob"
rule 13   input: "rob"
rule  5   input: "rob"
rule  5 returns: "rob"
rule 13 returns: "rmtc" "." "Central" "!" "rob"
rule  4   input: "rmtc" "." "Central" "!" "rob"
rule  9   input: "rmtc" "." "Central" "!" "rob"
rule  9 returns: "rmtc" "." "Central" "!" "rob"
rule  4 returns: "rmtc" "." "Central" "!" "rob"
```

Easier Address Verification

- Note that you can use more than one rule by entering a comma separated list as the first field of input for address test mode

- Nevertheless, the output from test mode is difficult to read and error prone for input

- Maybe a bit of UNIX toolsmithing could help. If we took the output above and did this:

```
% sed -n -e 's/.*returns: *//' \
      -e '/\$#/s/" "//gp' ...
```

we'd get

```
$# "ether" $@ "sun-barr.Sun.COM"
      $: "kolstad@usenix.ORG"
```

which is a bit more helpful, I think

Great Address Verification

- It almost begs for a **perl** script doesn't it?
  - I wrote one
  - It's yours, for free
  - It's fast (2.7u 3.7s 0:06 on Sun-4)
  - All modesty aside, it's WUNderful
  - Get it at boulder.colorado.edu as **~ftp/pub/sendmail.test.shar**

- It reads an input file of test addresses
- It resolves them, remembering which mailers were used
- It then applies those mailers' address rewriting rules as both senders and recipients

- All the while reporting the output to you (full sample run is in the appendices)

Verification Suite Output

- First, it tells you all the info you need to know to reproduce the results:

```
system: rmtc
current dir: /home/rmtc/rob/sendmail
config file: /etc/sendmail.cf
resolve file: address.resolve
towhom file: address.resolve
fromwhom file: address.resolve
```

- Then it resolves mail addresses; here's a typical line of output:

```
user@summit.central.sun.com --(ether )-->
    user@summit.central.Sun.COM[snail.Corp.Sun.COM]
```

- First item is the input address
- Second item is the resolved transport mechanism
- Third item is the remote address [and remote site]

More Verification

- It's fairly easy to verify the output of the resolutions by eye

- Then, the suite reports the recipient address transformations for each mailer used

- Finally, the suite reports the sender address transformations for each mailer used; sample output:

```
user -->  user@rmtc.Central.Sun.COM
```

- You can KNOW that your new mailer will work before you ever have to install it (assuming you cloned a working one)

Hiding Hosts

- To hide a hostname, consider this:
  - Up front, include a class of machine names to hide:
    CH hide1 hide2 hide3
  - In S0 (right after resolve local hostnames to the local mailer), hide the name:

```
R$*<$*$=H.LOCAL>$* $1<$2LOCAL>$4  hiddenhost.LOCAL
R$*<$*$=H.uucp>$*  $1<$2LOCAL>$4  hiddenhost.uucp
R$*<$*$=H>$*       $1<$2LOCAL>$4  hiddenhost
```

- Test it with the verifier, of course

A Note on Error Mailers

- Note that you can specify a mailer as ‘Error’ or some other undefined entity and pass back a short message to the originator of some mail.
- Resolved to ‘Error’ and specify funky names and addresses; **sendmail** ships the mail back to the sender along with an ‘invalid mailer’ error

Intense Debugging

- You can turn on detailed debugging with a command-line switch: **-d21.xx**
  - **-d** flag to **sendmail** specifies debug mode
  - level 21 specifies rulesets
  - range xx from 0-99 specifies verboseness
- Can produce reams of output:
  - **-d21.99** produces 686 lines of output for address blia!eric
  - **-d21.15** good level, produces 210 lines of output
  - **-d21.0** Turns debugging off

Choosing Debug Levels

- With debugging off, regular output in test mode looks like this:

```
% /usr/lib/sendmail -bt -CNewConfFile.cf -d21.0
ADDRESS TEST MODE
Enter <ruleset> <address>
> 0 kolstad@usenix.org
rule 3   input: "kolstad" "@" "usenix" "." "org"
rule 8   input: "kolstad" "@" "usenix" "." "org"
rule 8 returns: "kolstad" "@" "usenix" "." "org"
rule 6   input: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 6 returns: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 3 returns: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 0   input: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 3   input: "kolstad" "@" "usenix" "." "org"
rule 8   input: "kolstad" "@" "usenix" "." "org"
rule 8 returns: "kolstad" "@" "usenix" "." "org"
rule 6   input: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 6 returns: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 3 returns: "kolstad" "<" "@" "usenix" "." "org" ">"
rule 0 returns: $# "ether" $@ "sun-barr" "." "Ebay" "."
     "Sun" "." "COM" $: "kolstad" "@" "usenix" "." "ORG"
```

Higher Debug Levels

- Each token match shown in gory detail in high levels of debugging (-d21.15)
- With moderate debugging (kolstad@usenix.org, just a couple rules)

```
% /usr/lib/sendmail -bt -CNewConfFile.cf -d21.15
Version 4.1
ADDRESS TEST MODE
Enter <ruleset> <address>
> 0 kolstad@usenix.org
rewrite: ruleset  3   input: "kolstad" "@" "usenix" "." "org"
-----trying rule: $* "<" $+ ">" $*
----- rule fails
-----trying rule: $* "<" $* ">" $*
----- rule fails
-----trying rule: "@" $+ "," $+
----- rule fails
-----trying rule: "@" $+ ":" $+
----- rule fails
-----trying rule: $+
-----rule matches: $: $> "8" $1
$1: f7fff508="kolstad" f7fff510="@" f7fff512="usenix" f7fff519="."
                  f7fff51b="org"
-----callsubr 8
rewrite: ruleset  8   input: "kolstad" "@" "usenix" "." "org"
rewritten as: "kolstad" "@" "usenix" "." "org"
```


Serious Debugging

- With heavy debugging (same address, -d21.99), you get lots of output; here's the first few lines:

```
% /usr/lib/sendmail -bt -CNewConfFile.cf -d21.99
rewrite: ruleset  3   input: "kolstad" "@" "usenix" "." "org"
-----trying rule: $* "<" $+ ">" $*
ap="kolstad", rp=$*
ap="kolstad", rp="<"
ap="@", rp="<"
ap="usenix", rp="<"
ap=".", rp="<"
ap="org", rp="<"
ap=<null>, rp="<"
----- rule fails
-----trying rule: $+ "at" $+
ap="kolstad", rp=$+
ap="@", rp="at"
ap="usenix", rp="at"
ap=".", rp="at"
ap="org", rp="at"
ap=<null>, rp="at"
----- rule fails
-----trying rule: $* "<" $* ">" $*
ap="kolstad", rp=$*
ap="kolstad", rp="<"
ap="@", rp="<"
ap="usenix", rp="<"
```


Verbose Mode in Mail

- Run `mail` in verbose mode to watch its behavior in real time:

```
% /usr/ucb/mail -v evi@okeeffe.berkeley.edu
Subject: hi
.
Cc:
Null message body; hope that's ok
evi@okeeffe.berkeley.edu... Connecting to
                    okeeffe.berkeley.edu.nonlclsmtp...
220 okeeffe.Berkeley.EDU Sendmail 5.60/1.29 ready
                    at Tue, 30 Aug 88 17:37:02 PDT
>>> HELO boulder.Colorado.EDU
250 okeeffe.Berkeley.EDU Hello boulder.Colorado.EDU,
                    pleased to meet you
>>> MAIL From:<evi@boulder.Colorado.EDU>
250 <evi@boulder.Colorado.EDU>... Sender ok
>>> RCPT To:<evi@okeeffe.berkeley.edu>
250 <evi@okeeffe.berkeley.edu>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 Ok
>>> QUIT
221 okeeffe.Berkeley.EDU closing connection
evi@okeeffe.berkeley.edu... Sent
```


Speak to `sendmail` in its own tongue (`SMTP`)

- `SMTP` – Simple Mail Transport Protocol
  - `telnet` to port 25
  - Simple command set, only 14 instructions

```
HELO - to start a conversation
MAIL From: <reverse-path> - initiate a mail
              transaction, identify sender
RCPT To: <forward-path> - identify recipient
DATA - begin the message, terminate with a "."
QUIT - to end the conversation
HELP - get help
VRFY - to check addresses and aliases
```

- Note that VRFY and its cousins are security holes for aliases

Starting `sendmail`

- Modes and Common Options
  - **-bd** Daemon Mode – listen on port 25 for work
  - **-bz** Freeze the Configuration File – pre-process names in the config file to improve performance
  - **-q1h** Process the Queue – process every hour (1h) (for retrying failed deliveries)
  - **-q15m** Process the Queue – process every 15 minutes
  - **-bt** Test Mode – test rewriting rules
  - **-Cconfig_file_name** – use specified config file
- Usually started in `/etc/rc.local` Startup Script
- **sendmail** uses **-bp** for **mailq** and **-bi** for **newaliases**

Installing a New Config File

- Must restart **sendmail** for changes to take effect:

```
# cd /usr/lib
# mv sendmail.cf sendmail.cf-
# cp sendmail.new sendmail.cf
# /usr/lib/sendmail -bz
# ps ax | grep sendmail
# kill 753    (or whatever)
# /usr/lib/sendmail -bd -q1h
#
```

- For speed, freeze the configuration file with the **-bz** flag before restarting **sendmail**
- Watch for vendor upgrades overwriting your config files

`sendmail` and Security

- Impersonation is easy, beware:
  - Requests for master keys for undergrads
  - Grade changes from F to A
  - Salary adjustments from minimum wage to presidential pay
- Bad history, Internet worm/virus attacked Vaxen/Suns
  - **sendmail** 5.61 for BSD is claimed to be wormproof
  - uunet now has public domain source for sendmail in ~ftp/bsd-sources/src/network/sendmail.tar.Z

Mail Aliases – `/usr/lib/aliases`

- Global aliasing or renaming mechanism
- Can be nested, detects loops even between machines
- Supports mailing lists
- Can mail to a program
- Must define the **postmaster** alias to local mail wizard
- **newaliases** program rebuilds global alias database

Alias Format

Format of alias entry depends on the file

- In **/usr/lib/aliases** – global aliases for the whole machine
```
alias-name: real names[s]
      nemeth: evi
      speakers: rob,trent,evi
      postmaster: trent,coggs
```
- In **.mailrc** – a user's personal aliases
```
alias new-name old-name[s]:
     alias nemeth evi
     alias lisa evi kolstad polk trent
```
- In **.forward** – reroute a user's personal mail
```
name1, name2, ...   evi, evi@okeeffe.berkeley.edu
```
- **\user** means "straight to user's mailbox"

Network Global Aliases

- Easy to maintain – change in one place and distribute
- Use files to make simultaneously global/Local
- Trouble mail example:
```
trouble: :include:/usr/local/adm/trouble.alias
          (contains tmr, rob, jeff)
troubletrap: "/usr/local/adm/logs/troublemail"
tmr: troubletrap@boulder,trent,evi
```
  - Includes local file, tailored to each machine
  - Includes global log file on machine boulder
  - Includes global trouble mail readers
  - For this machine: trent, evi, rob, jeff
  - For all machines: trent, evi, xxx[s]

Reading Mail Headers

- Sample mail message header (line# is not part of header)
```
 1:  From evi@lafayette.dartmouth.edu Mon May 21 21:43:25 1990
 2:  Received: from Central.Sun.COM (texsun.Central.Sun.COM)
 3:          by rmtc.Central.Sun.COM (4.1/4.7)
 4:          id AA19584; Mon, 21 May 90 21:43:22 MDT
 5:  Received: from Sun.COM (sun-barr.EBay.Sun.COM)
 6:          by Central.Sun.COM (4.1/SMI-4.1)
 7:          id AA01883; Mon, 21 May 90 22:43:52 CDT
 8:  Received: from lafayette.dartmouth.edu by Sun.COM
 9:          (4.1/SMI-4.1) id AA09904; Mon, 21 May 90
10:          20:43:51 PDT
11:  Received: by lafayette.dartmouth.edu (5.61a+YP/4.1)
12:          id AA00522; Mon, 21 May 90 23:43:34 -0400
13:  Date: Mon, 21 May 90 23:43:34 -0400
14:  From: evi@lafayette.dartmouth.edu (Evi Nemeth)
15:  Message-Id: <9005220343.AA00522@lafayette.dartmouth.edu>
16:  To: kolstad@rmtc.Central.Sun.COM
17:  Subject: more on sysadmin notes
18:  Cc: evi@lafayette.dartmouth.edu
```


Message Routing:

```
line 11: From machine lafayette.dartmouth.edu (origination machine)
line  8: Sun.COM gets message from lafayette
line  5: Central.Sun.COM gets message from Sun
line  2: rmtc.Central.Sun.COM gets message from Central
```

One Mail System Design

- Central hub machine and forced cooperation among machines
- Cooperating secondary machines (like workstations) follow a global paradigm of mail use
- Users obey a global paradigm of aliases and mail residence
  - Problem if politics is an issue
  - Problem if server queue grows too long
- Master hub machine coordinates global paradigms
- (Remember: Increasing flexibility eventually makes administrator's lives into living nightmares)
- Mount `/usr/spool/mail` so workstation users can read their mail there

One Mail System Design: Machines

- Assume multiple machines on network
- Designate one hub machine as gateway
  - Run smart mail configuration here
  - Understands connections to the outside world
- Other machines run generic mail configuration
  - Easier to maintain
  - Possible loss of efficiency, may not care
  - Bad if hub machine down
- Strategy – find a configuration file that works and localize it

One Mail System Design: Users

- User has only one login name
- User has only one filesystem to read mail (mail home)
  - Done with global **aliases** file
  - May NFS mount **/usr/spool/mail** for reading (otherwise there can be locking problems)
  - Local changes maintained by sysadmin via **aliases** file (infrequent)
  - Permanent changes to remote machine also done by sysadmin (also usually infrequent)
  - Temporary change to remote machine done by user in **.forward** file under user's control; forwarding to local alias for the same user file may create an alias loop

Master or Hub Mail Machine

- Should know about network topology
- Should support external connections
  - Local **DECNET** (if any)
  - **BITNET**
  - **CSNET**
  - Internet
  - **UUCP**
- Probably runs the name server (**BIND**, **named**)
- Might run **pathalias** to expand **uucp** addresses (or subscribe to uunet)
- Needs plenty of spooling space
- Becomes a bottleneck if down long

Pre-defined Configuration File Variables

a	Origination date of the message (Arpanet Format)
b	Current date
c	Hop count
d	Date in UNIX format
f	Sender address
g	Sender address relative to the recipient
h	Recipient host
i	Queue id
p	Sendmail's pid
r	Protocol used
s	Sender's host name
t	Numeric representation of the current time
u	Recipient user
v	Version number of sendmail
w	This site's hostname
x	Full name of the sender
y	Id of the sender's teletype
z	Home directory of the recipient

For More Information

- See the Tutorials:
  SMM: Sendmail Installation and Operation Guide
  SMM: Sendmail – An Internetwork Mail Router
- *UNIX System Administration Handbook*, Prentice-Hall, Inc. 1989. Nemeth, Snyder, and Seebass.
- Reference Commands:

aliases.5	mail.1	rmail.8
biff.1	mailaddr.7	sendmail.8
binmail.1	mconnect.8	talk.1
comsat.8c	newaliases.8	xsend.1
from.1	prmail.1	

- Newsgroups: **comp.mail.sendmail**

DNS

Why use the Domain Name System

- DNS is a solution to the difficult problem of name resolution
- It works
- RFC 1123 (Requirement for Internet Hosts) states:

  Every Internet host MUST implement a resolver for the Domain Name System (DNS), and it MUST implement a mechanism using this DNS resolver to convert host names to IP addresses and vice-versa.

Need for Domain Names

- Need to have a mapping between
  - Hostnames `ucbvax.berkeley.edu`
  - Internet addr's 128.32.133.1
- Early 80's hosts on the Internet registered with the Network Information Center (NIC) and were listed in a global table
- The size of the table and the frequency of updates were nearing the limit of manageability
- Additionally, an increasingly large and irregular set of methods for identifying the location of a mailbox had evolved

More Need For Domain Names

- Network population was changing in character
- The applications on the Internet were getting more sophisticated and creating a need for general purpose name service

DNS Design Goals

- Consistent name space
- Maintained in a distributed manner
- The source site controls the tradeoffs between the cost of acquiring data, the speed of updates, and the accuracy of caches
- The facility be generally useful

More DNS Design Goals

- Ability to use the same name space with different protocol families
- Name server's transactions should be independent of the communications system that carries them (data grams vs. virtual circuits).
- The system should be useful across a wide spectrum of host capabilities

Assumptions About Usage

- The System Administrator provides a configuration file with:
  - The definition of zone boundaries
  - Master files of data
  - Updates to master files
  - Statements of the refresh policies desired
- The Domain Name System Provides:
  - Standard formats for resource data
  - Standard methods for querying the database
  - Standard methods for name servers to refresh local data from foreign name servers

Elements of the DNS

- The DNS has 3 major components:
  - The Domain Name Space and Resource Records (RR)
  - Name Servers
  - Resolvers

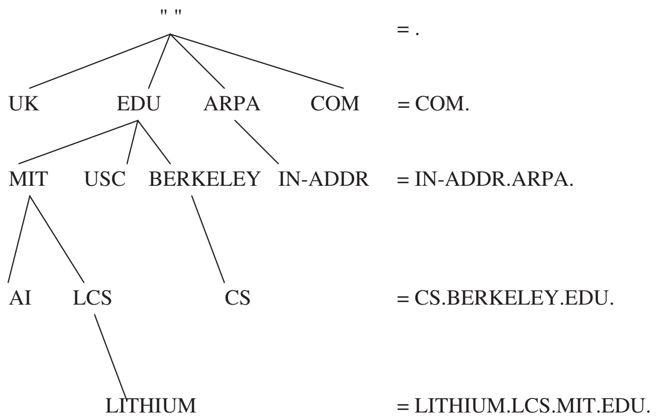
Domain Name Space and Resource Records

- Specifications for a tree structured name space and data associated with the names
- Conceptually, each node and leaf of the domain name space tree names a set of information (RRs)
- Query operations are attempts to extract specific types of information from a particular set
- A query names the domain name of interest and describes the type of resource information that is desired

Resource Records (RRs)

- The set of resource information associated with a particular name is composed of separate Resource Records (RRs)
- The Resource Records are the data contained in the name server data files

Example Name Space



Name Server

- Name servers are the repositories of information that compose the domain database (information stored as RRs)
- The database is partitioned into sections called zones which are distributed among the name servers (usually divided along organizational boundaries)
- The essential task of a name server is to answer queries using data in its zone

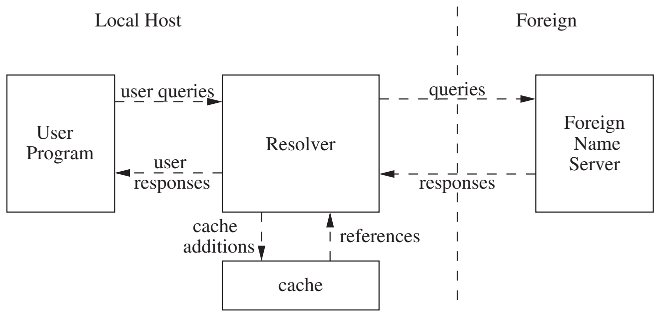
Queries to the Name Server

- Name servers can answer queries with the answer to the query or a referral to another name server (iterative approach)
- Each zone must have at least two name servers
- A name server is "authoritative" for the parts of the domain tree for which it has complete information

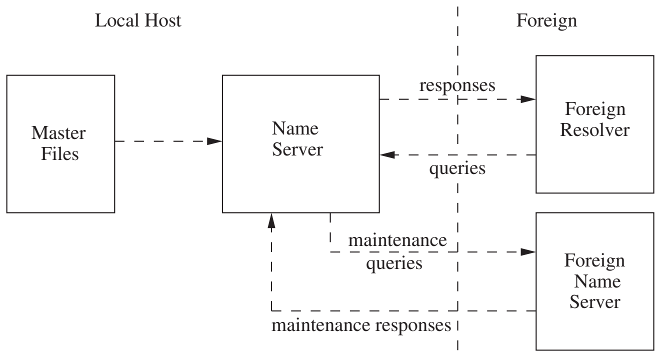
Resolver

- Programs that interface user programs to domain name servers are "resolvers"
- Resolver's goal is elimination of network delay and name server load from most requests by answering them from its cache of prior results

Common Configuration I



Common Configuration, II



DNS Implemented

- The Berkeley Internet Name Domain (BIND) server implements the DARPA Internet name server for the UNIX operating system
- It is the most complete and widely used implementation of domain name servers and resolvers – people often use BIND and DNS as synonyms
- A complete copy of BIND and documentation, including the BIND Operations Guide (BOG), can be obtained via anonymous ftp from okeefe.cs.berkeley.edu

BIND Composition

- BIND includes two parts:
  - The resolver (which consists of a group of routines that reside in the C library, /lib/libc.a)
  - The name server, **named**, which is a daemon that runs in the background and services queries
- There are six files that must be created and maintained to operate BIND, they are:

Name Server Files	Resolver File
`named.boot`	`resolv.conf`
`named.local`	
`hosts`	
`hosts.rev`	
`root.cache`	

Name Server File Information

- `.` A dot in the name field refers to the current domain
- `@` A @ denotes the current origin
- `..` Two free standing dots represent the null domain names of the root when used in the name field
- `\X` Where `X` is any character other than a digit, quotes that character so that its special meaning does not apply
- `\DDD` where each `D` is a digit, is the octet corresponding to the decimal number described by `DDD`

More Name Server File Information

- `()` Parentheses group data that crosses a line
- `;` Semicolon starts a comment; the remainder of the line is ignored
- `*` An asterisk signifies wildcarding

More Information on Name Server Files

- Some special directives in column 1 perform special functions
- Inclusion:
  $INCLUDE filename
  will load the file named filename
- Used to store different groups of data in different files, e.g., machine types, or mailbox information
- Origin:
  $ORIGIN domain_origin
  changes the origin in a data file
- Used to put more than one domain in a single file

resolv.conf File

- This file provides a list of possible name serving machines to which queries should be directed

  Example resolv.conf:

```
domain          our.domain
nameserver      128.115.32.7
nameserver      128.115.32.6
```


named.boot File

- The boostrap file for **named**, which is read when **named** starts
- Specific to **BIND**
- Tells **named** what type of name server it is, which zones it has authority over, and where to get its initial data
- Located in /etc/named.boot
  - Location can be changed at **named** compile time
  - Location can be changed on the command line when named is started (usually from /etc/rc.local file at boot time)

named.boot File: Other Elements

- The directory line specifies the directory in which the name server should run, allowing the other file names in the boot file to use relative path names
  directory /usr/etc/nameserver
- The line in the file that designates the server as a primary server for a zone looks as follows:
  primary Berkeley.EDU hosts
- First field specifies that the server is primary for the zone stated in the second field
- The third field is the name of the file from which the data for the zone is read

Types of Domain Name Servers

- Master Servers
  - Primary
  - Secondary
- Caching-Only Server
- Forwarding/Slave Server
- Remote (Client) ''Server''

Master Servers

- Each zone must have at least two master name servers that maintain all the data corresponding to the zone, making them the authority for that zone
- The primary master server loads the master copy of its data from the data files on disk when named is started
- The primary master server may also delegate authority to other servers in its zone, as well as to servers outside of it

Secondary Master Servers

- Secondary master servers request the data for the zone from the primary master server
- The secondary master server periodically checks with the primary master server to see if it needs to update its data
- A server may act as the primary server for multiple zones, and as the secondary server for multiple zones
- The line in the named.boot file that designates the server as a secondary server for a zone looks as follows:

```
secondary Berkeley.EDU  128.32.0.4  hosts.bak
```


Other Types of Domain Name Servers

- Caching-only server does not maintain authoritative information for any of its responses
- Forwarding servers handle requests by forwarding them to other servers
- The line in the named.boot file that designates the server as a forwarding server for a zone looks as follows:

```
forwarders  128.32.0.4  128.32.0.10
slave
```

More named.boot File

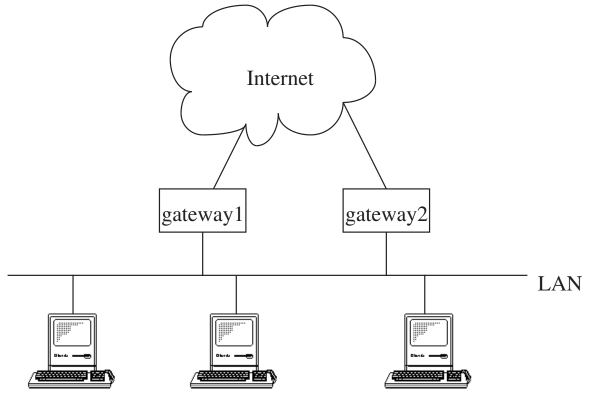
- All name servers should have the following line in the named.boot file to prime the name server's cache with the names of the root servers

  `cache   .   root.cache`

- The third field is the name of the file that contains the names of the root servers

- You can ftp the current list of root servers from internic.net from: <NETINFO:ROOT-SERVERS.TXT>

our.domain Master Servers



Example named.boot File

```
; boot file for primary name server
;
; type          domain               source file or host
;
directory                            /usr/etc/ns
;
primary our.domain                   hosts
;
primary 115.128.IN-ADDR.ARPA         hosts.rev
;
primary 0.0.127.IN-ADDR.ARPA         named.local
;
cache           .                    root.cache
```


BIND Data Files

- There are 3 standard files for specifying the data for a domain:
  - named.local
  - hosts
  - hosts.rev

- These files use the standard Resource Record format to store the authoritative data for the zone

- Basically, these are the files that have all the information about the hosts in this domain

Standard Resource Record Format

- All RRs have the same top level format
- The standard RR format is specified in RFC 1035
- The general description of these records is:

```
{name}   {ttl}   address class
      record type   record specific data
```

- *Description folded for display purposes*

The (optional) name

```
{name}   {ttl}   address_class
      record type   record specific data
```

- The first field is the optional name of the domain record
- It must always start in column 1
- It may be left blank for some RRs; in that case it defaults to the name of the previous RR

The Time to Live

- Name server information is dynamic!
- TTL field allows you to weigh cost of name server performance vs. data accuracy
  - Small TTL insures accurate data across the network
  - Small TTL increases load on your name servers
- TTLs can be changed for individual RRs as well
  - Useful if you plan to relocate a host

Time to Live Format

```
{name}   {ttl}   address_class
      record type   record specific data
```

- The second field is the optional time-to-live (ttl) field
- Specifies how long any name server is allowed to store (cache) the data
- Blank ttl field defaults to the ttl specified in the Start Of Authority (SOA) RR

The address_class Field

```
{name}   {ttl}   address_class
      record type   record specific data
```

- The third field is the address_ class code
- Current codes are:

Abbr.	Facility
IN	Internet
CH	Chaosnet
HS	Hesiod

The type Field

- Fourth field is the RR type
- The field(s) after that are dependent on the type field

Type	Value and Meaning
SOA	start of a zone of authority
NS	authoritative name server
A	host address
CNAME	canonical name for an alias
WKS	well known service
HINFO	host information
PTR	domain name pointer
MX	mail exchange
TXT	text strings

Example SOA Record

- SOA: Start of a zone of authority

```
name ttl class type  data
@        IN    SOA   bt.mit.edu. postmaster.bt.mit.edu. (
                     101          ; Serial
                     10800        ; Refresh after 3 hours
                     3600         ; Retry after 1 hour
                     604800       ; Expire after 1 week
                     86400 )      ; Minimum TTL of 1 day
```


Example Address Record

- A: address

```
name    ttl class type data
ucbvax      IN    A    128.32.133.1
```

- The address record lists the address for a given machine
- The name field is the machine name and the address is the network address
- There should be one A record for each address of the machine

Example Explicit TTL

- TTL on an individual resource record

```
name    ttl     class type data
ucbvax  1800    IN    A    128.32.133.1
```

- TTL field in the SAO record is the default minimum
- TTL on an individual RR will allow smaller minimum

Example NS record

- NS: authoritative name server

```
name ttl class type data
         IN    NS   ucbvax.Berkeley.EDU.
```

- The name server record lists a name server responsible for a given domain
- The first name field lists the domain that is serviced by the listed name server
- One NS record for each Primary Master Server for the domain

More Example NS Record

```
name  ttl class type data
          IN    NS   sri.com.
          IN    NS   kl.sri.com.
          IN    A    10.1.0.2
```

- If the name server host for a particular domain is itself inside the domain, then a "glue" record is needed
- A "glue" record is an A RR that specifies the address of the server
- Glue records are only needed in the server delegating the domain, not in the domain itself

Example Host Information Records

- HINFO: host information

```
name    ttl class type   data
            IN    HINFO  VAX-11/780 UNIX
```

- Host information record, HINFO, is for host specific data
- It lists the hardware and operating system that are running at the listed host
- Only a single space separates the hardware information and the operating system information.
- Exactly one HINFO record for each host

Example Well Known Service Records

- WKS: Well known service

```
name ttl class type data
         IN    WKS  128.32.133.1 UDP who timed
         IN    WKS  128.32.133.1 TCP (telnet
                                 smtp pop finger)
```

- The well known services record lists services supported by a particular protocol at a specified address
- The list of services and port numbers comes from the list of services specified in /etc/services
- Only one WKS record per protocol per address

Example Canonical Name Records

- CNAME: Canonical name

```
name    ttl class type  data
ucbvax      IN    CNAME monet
```

- The Canonical Name record specifies an alias for a canonical name.
  - Should be the only record associated with the alias name
  - All other resource records should be associated with the canonical name – not with the alias
  - Records that include a domain name as their value (e.g., NS, MX) should list the canonical name, not the alias

Example Mail Exchanger Records

- MX: Mail exchanger

```
name    ttl class type data
            IN    MX   0   ucbvax
            IN    MX   100 ucbjade
            IN    A    128.32.133.1
```

- Mail exchange records specify a machine that knows how to deliver mail to a domain or machines in a domain.
  - More than one MX record for a given name is OK
  - Preference value indicates order mailer should follow when multiple delivery paths exist
  - Higher value → lower the preference
  - 0 is highest preference

More MX Records

```
name   ttl class type data
foo.com.     IN   MX   10 RELAY.CS.NET.
*.foo.com.   IN   MX   20 RELAY.CS.NET.
```

- Use names with the wildcard ('*') for mail routing with MX records
- The wildcard resource record above states that the mail exchanger for *.foo.com is RELAY.CS.NET
- It is a good idea to include an A record with your MX records for older mailers (see previous page)

Example Text Strings Record (Hesiod)

- TXT: Text strings

```
name  ttl class type data
juser       HS   TXT
          "juser:*:32:11:J. User:/h/juser:/bin/csh"
juser       HS   TXT "POP po2.mit.edu juser"
```

- Hesiod extensions to BIND provide naming for services and data objects

named.local Data File

- This file specifies the address for the local loopback interface, also known as localhost, with the address 127.0.0.1
- The location of this files is specified in the boot file

Sample named.local data file

```
0.0.127.in-addr.arpa.
      IN  SOA  gw1.our.domain. postmaster.gw1.our.domain. (
                    215        ; Serial
                    10800      ; Refresh after 3 hours
                    1800       ; Retry after half an hour
                    604800     ; Expire after 1 week
                    86400   )  ; Minimum TTL of 1 day
      IN  NS   gw1.our.domain.
      IN  NS   gw2.our.domain.
1.0.0.127.in-addr.arpa.
      IN  PTR  localhost.
```

Lines folded for display purposes

hosts Data File

- This file contains all the data (RRs) about the machines in this zone
- The location of this file is specified in the named.boot file
- Sample:

```
     Configuration information for our.domain
@   IN   SOA  gw1.our.domain. postmaster.gw1.our.domain. (
                            215      ; Serial
                            10800    ; Refresh after 3 hours
                            1800     ; Retry after half an hour
                            604800   ; Expire after 1 week
                            86400 )  ; Minimum TTL of 1 day
    IN   NS   gw2.our.domain.
    IN   NS   gw1.our.domain.
    IN   MX   0 gw2.our.domain.
    IN   MX   10 gw1.our.domain.
    IN   A    128.115.32.7        ;for old mailers
```

More Sample File

```
localhost IN  A     127.0.0.1
;gateway1 IN  A     128.115.32.6
          IN  HINFO SUN-4 UNIX
          IN  WKS   128.115.32.6 UDP echo discard domain tftp sunrpc
          IN  WKS   128.115.32.6 TCP ( echo
                       discard systat daytime netstat
                       ftp telnet smtp time whois
                       finger sunrpc domain nameserver);
gateway2  IN  A     128.115.32.7
          IN  HINFO SUN-4 UNIX
          IN  WKS   128.115.32.7 UDP echo discard domain tftp sunrpc
          IN  WKS   128.115.32.7 TCP (   echo
                      discard systat daytime netstat
                      ftp telnet smtp time whois finger
                      sunrpc domain nameserver)
choke1  IN   A      128.115.0.4
        IN   A      128.115.16.4
        IN   A      128.115.32.4
        IN   A      128.12.18.50
        IN   HINFO  CISCO AGS
        IN   WKS    128.115.0.4   TCP ( tftp
                       telnet snmp echo systat)
mikl    IN   CNAME  choke1
rt1     IN   CNAME  choke1
```


Even More Sample File

```
choke2  IN   A       128.115.0.5
        IN   A       128.115.16.5
        IN   A       128.115.32.5
        IN   A       128.12.18.51
        IN   HINFO   CISCO AGS
        IN   WKS     128.115.0.5   TCP ( tftp
                        telnet snmp echo systat)
gabby   IN   CNAME   choke2
rt2     IN   CNAME   choke2


mordor  IN    A       128.115.0.3
        IN    HINFO   VAX-11/750 UNIX
        IN    WKS     128.115.0.3     UDP (
                  echo discard domain tftp sunrpc)
        IN   WKS     128.115.0.3   TCP ( echo
                         discard systat daytime
                         netstat ftp telnet smtp
                         time whois finger nntp )
```

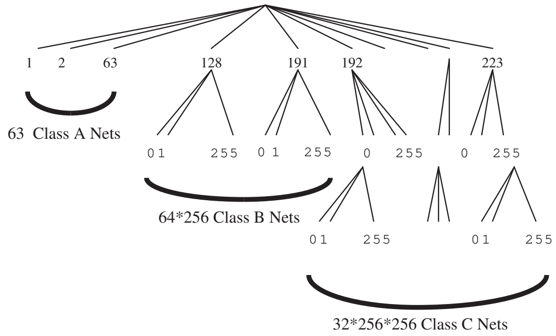

Final Part of Sample File

```
;
cyclops IN    A       128.115.17.16
        IN    HINFO   MICROVAX2 VMS
        IN    WKS     128.15.17.16  TCP (   ftp
                         telnet smtp )
;
peter   IN    A       128.115.32.8
        IN    HINFO   TELEBIT NetBlazer
        IN    WKS     128.115.32.8  TCP (   tftp
                         telnet snmp echo systat slip )
;
```

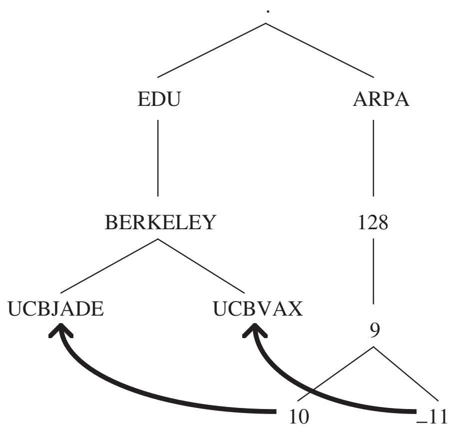

hosts.rev Data File

- This file specifies the IN-ADDR.ARPA domain
- This is a special domain which was created to allow address to name mapping (or inverse mapping):
  `128.32.133.1` ↔ `ucbvax.berkeley.edu`
- To translate IP address a.b.c.d to its corresponding host name,
  - Reverse the octets: a.b.c.d → d.c.b.a
  - Add the IN-ADDR.ARPA. suffix: d.c.b.a.IN-ADDR.ARPA.
- Query name server with a type=PTR, class=IN and Name=1.133.32.128.IN-ADDR.ARPA to see ucbvax.Berkeley.EDU

IN-ADDR.ARPA Domain



Example IN-ADDR.ARPA. PTR



Example Domain Name Pointer Records

- **PTR: Domain name pointer (required for your installation)**

  ```
  name  ttl class type data
  1.133     IN    PTR  ucbvax.Berkeley.EDU
  ```
- **The Domain Name Pointer record allows special names to point to some other location in the domain**
- **The example is used in setting up the reverse pointer for the special IN-ADDR.ARPA domain (for address resolution)**

Example hosts.rev Data File

```
;configuration information; hosts.rev for our.domain
@     IN  SOA  gw1.our.domain. postmaster.gw1.our.domain. (
                     215      ; Serial
                     10800    ; Refresh after 3 hours
                     1800     ; Retry after half an hour
                     604800   ; Expire after 1 week
                     86400 )  ; Minimum TTL of 1 day
      IN NS    gw1.our.domain.
      IN NS    gw2.our.domain.
6.32  IN PTR   gw1.our.domain.
7.32  IN PTR   gw2.our.domain.
4.0   IN PTR   choke1.our.domain.
4.16  IN PTR   choke1.our.domain.
4.32  IN PTR   choke1.our.domain.
50.18 IN PTR   choke1.our.domain.
5.0   IN PTR   choke2.our.domain.
5.16  IN PTR   choke2.our.domain.
5.32  IN PTR   choke2.our.domain.
51.18 IN PTR   choke2.our.domain.
3.0   IN PTR   mordor.our.domain.
16.17 IN PTR   cylops.our.domain.
8.32  IN PTR   peter.our.domain.
;
1.0.0.127.in-addr.arpa. IN  PTR localhost.
```

root.cache File

- The name server needs to know the servers that are authoritative name servers for the root domain of the network
- To do this we prime the name server's cache with their addresses

Example root.cache File

```
; initial cache data for root domain servers
        99999999   IN  NS    KAVA.NISC.SRI.COM
        99999999   IN  NS    AGS.BRL.MIL
        99999999   IN  NS    C.NYSER.NET
        99999999   IN  NS    NIC.NORDU.NET
        99999999   IN  NS    TERP.UMD.EDU
        99999999   IN  NS    NS.NASA.GOV
;
; prep the cache (hot-wire the addresses)
;
NS.NIC.DDN.MIL     99999999  IN  A   192.112.36.4
KAVA.NISC.SRI.COM  99999999  IN  A   192.33.33.24
AGS.BRL.MIL        99999999  IN  A   26.3.0.29
C.NYSER.NET        99999999  IN  A   192.33.4.12
NIC.NORDU.NET      99999999  IN  A   192.36.148.17
TERP.UMD.EDU       99999999  IN  A   128.8.10.90
NS.NASA.GOV        99999999  IN  A   128.102.16.10
```


Setting Up Your Own Domain

- Request a form to establish or join a domain from the NIC
  - You can send mail to hostmaster@internic.net
  - Or file netinfo:domain-template.txt (via anon `ftp`) from internic.net
  - Network Solns./InterNIC Registration Service
    505 Huntmar Park Dr. Herndon, VA 22070
    (800) 444-4345
- Other help at internic.net (via anon `ftp`):
  - netinfo:domains.txt – table of top level domains
  - netinfo:domain-info.txt –list of top level and 2nd level domains
  - netinfo:domain-contacts.txt – top level and 2nd level domains and their (human) contacts

Registering With Your Parent

- DARPA INTERNET – contact hostmaster@internic.net
  - Use `whois` to see if your network is registered with the NIC, e.g.:
    `#` *whois -h internic.net net 128*
    (This version of whois is better than that supplied on Suns)
- CSNET – contact the Coordination and Information Center (CIC), cic@sh.cs.net
- BITNET – contact info@bitnet

Debugging and Maintaining Your Domain

- **nslookup** is an interface that enables you to query name servers
- Dump the name server database to /usr/tmp/named_dump.db or /var/tmp/named_dump.db with the INT signal
- **doc** is used to diagnose DNS domains, ftp it from: venera.isi.edu
- **namedconf** is a locally developed tool that aids in creating and maintaining the BIND data files

nslookup Output

```
gate2.our.domain% nslookup
Default Server:  our.domain
Address:  128.115.32.7
> mit.edu
Server:  our.domain
Address:  128.115.32.7
Name:    mit.edu
Address:  18.72.2.1
> set querytype=PTR
> 1.2.72.18.in-addr.arpa.
Server:  our.domain
Address:  128.115.32.7
1.2.72.18.in-addr.arpa
       host name = MIT.MIT.EDU
> sgi.com
Server:  our.domain
Address:  128.115.32.7
Non-authoritative answer:
Name:    sgi.com
Address:  192.48.153.1
```


More nslookup

```
> server sgi.com
Default Server:  sgi.com
Address:  192.48.153.1
> sgi.com
Server:  sgi.com
Address:  192.48.153.1
Name:    sgi.com
Address:  192.48.153.1
> server our.domain
Default Server:  our.domain
Address:  128.115.32.7
> set querytype=SOA
```


Last Part of nslookup

```
> set all
Default Server:  our.domain
Address:  128.115.32.7
Set options:
 nodebug         defname  search    recurse novc
 querytype=SOA   class=IN timeout=6 retry=4
 domain=our.domain
 search list: our.domain
 root=sri-nic.arpa.
> mit.edu
Server:  our.domain
Address:  128.115.32.7
mit.edu origin = BITSY.MIT.EDU
   mail addr = NETWORK-REQUEST.MIT.EDU
   serial=477, refresh=3600, retry=300,
         expire=3600000, min=21600
> ^D
gate2.our.domain%
```

Symptoms

- Local name can't be looked up
- Remote names can't be looked up
- Wrong or inconsistent answer
- Lookups take a long time
- rlogin and rsh to host fails access check
- Access to services denied
- Name server is infected with bogus root server data
- Name server keeps loading old data

Scaling

- Note that DNS and BIND scale very well (to millions of machines)

Where To Get Additional Information

- Name Server Operations Guide for BIND, UC Berkeley
- "DNS and Bind" by P. Albitz and C. Liu, published by O'Reilly and Associates
- Request For Comments (RFCs):
  - 1033 Domain Administrators Operations Guide
  - 1034 Domain Names – Concepts and Facilities
  - 1035 Domain Names – Implementation and Specification
  - 1123 Requirements for Internet Hosts – Application and Support

Firewalls: Routers

The Problem

- Connecting to the Internet carries with it inherent security risks
  - Anyone can connect to your machine
  - They may not all be trustworthy
- Preventing security problems at each server and client is extremely difficult
  - Might be impossible
  - The Morris worm certainly opened some eyes
- Nevertheless, the packets must be stopped

A Solution: Firewalls

- One way to eliminate complex host and client security monitoring is "Firewalls"
  - The Firewall guards a site at its front door
  - Only packets "approved" by the firewall make it through
  - Security is dramatically enhanced
  - Protection in depth

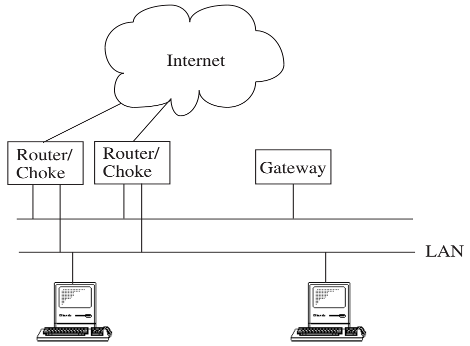
Firewall Approach

- Purpose – set up a configuration that puts up a wall between your local installation and the outside world
- Should be configured to allow certain functionality (`ftp`, `telnet`, `mail`)
- But, in general, make it difficult for an attacker on the outside to penetrate the firewall and gain access to your local nets

Some Firewall Nomenclature

- **Screening Router/Choke – have the ability to block traffic on an IP port level**
- **Bastion Host – a system with extra security, auditing, and modified software**
- **Screened Host Gateway – a screening router and a bastion host combination**
- **Screened Subnet – an isolated subnet situated between the Internet and the private network; also called the demilitarized zone (DMZ)**
- **Application Gateway – a forwarding service that runs across a firewall, usually run on a bastion host**

Firewall Network Example



Implementing the Firewall Design

- **Lock the front door: choking-router restricts routing (as above)**
- **Secure and empower the gateway machine**
  - **Standard host security on the gateway**
  - **Proxy commands to re-implement functionality**
  - **DNS**
  - **`sendmail`**

It's all Trade Offs!

- **There are variations on this theme**
- **You will need to identify the security and functionality requirements for your site**
- **Implement the firewall design that best suits your site**

Let's start with the Router

- Router/Choke
- Restricts Packets' Source/Destination
- Restricts Protocols

Choke-Specific Configuration

- Access lists restrict transmission to specified addresses, whether identifying server ports, hosts, or routers
- Packet-type control specifies which packets may pass, such as mail packets, file transfers, and remote logins

IP Access Lists

- An access list is a sequential collection of permit and deny conditions for Internet addresses
  - Match: appropriate permit/deny
  - No match: deny
- To specify an access condition an access list configuration command is used:

```
access-list list {permit|deny} address wildcard-mask
no access-list list
```


Router Configuration Warm-up

Address	Mask	Action
X	1	Ignore (so address is allowed)
0.0.0.0	255.255.255.255	Any address allowed
199.12.1.1	0.0.0.0	Only 199.12.1.1

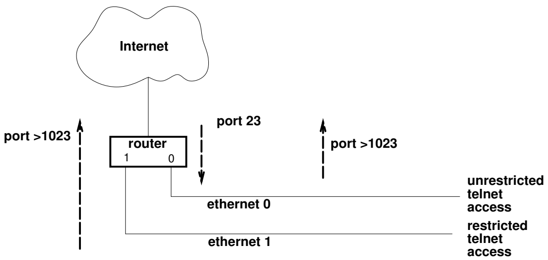
Choices, Choices, Choices!

- Security is the reciprocal of convenience
- Begin with a clear understanding of the site security policy and goals
- Each application function should be evaluated for design feasibility, potential risks, and user benefits
- The challenge becomes designing for the functionality which still provides the appropriate security
- Choose the right mix for your site!

Selective Filtering Approach

- Uses the router to screen traffic
- Need not involve a bastion host
- Can avoid addition of proxy application software on clients

Filtering Telnet Example



Router Configuration for Telnet Access

```
interface ether 0
ip address 128.115.32.0 255.255.255.0
ip access-group 102

ip access-list 102 permit tcp   0.0.0.0  255.255.255.255
128.115.32.0    0.0.0.255 eq 23
ip access-list 102 permit tcp   0.0.0.0  255.255.255.255
128.115.32.0    0.0.0.255 gt 1023

interface ether 1
ip address 128.115.0.0 255.255.255.0
ip access-group 103

ip access-list 103 permit tcp   0.0.0.0  255.255.255.255
128.115.0.0     0.0.0.255 gt 1023
```

Telnet Configuration Trade Offs

- **Can leave you wide open above port 1023**
- **Router configuration can get to be a hit/miss proposition**
- **Suggest telnet outbound only**
- **A compromise might be:**
  - **telnet inbound from static addresses only**
  - **Force telnet through a bastion host**
  - **Use a terminal server**

Monitoring Selectively Filtering Firewalls

- **Servers can run on your net without your knowledge**
- **probe_tcp_ports is a tool to look for services on the net**
  - **Run it via cron to monitor your net**
  - **Posted to firewalls@greatcircle.com**
- **Output gives port numbers and service names:**

```
Host gateway.com, Port 23 ("telnet" service)
                    connection ... open.
Host gateway.com, Port 25 ("smtp" service)
                    connection ... open.
Host gateway.com, Port 53 ("domain" service)
                    connection ... open.
```


Additional Router Configuration

- telnet from static address only

```
ip access-list 102 permit tcp  192.25.36.1  0.0.0.0
128.115.32.0    0.0.0.255 eq 23
ip access-list 102 permit tcp  0.0.0.0  255.255.255.255
128.115.32.0    0.0.0.255 gt 1023
```

- Forcing telnet through a bastion host

```
ip access-list 103 permit tcp  128.115.32.6   0.0.0.0
128.115.0.0   0.0.0.255 eq 23
ip access-list 103 permit tcp  128.115.32.6   0.0.0.0
128.115.0.0   0.0.0.255 gt 1023
```


Commonly Filtered Ports

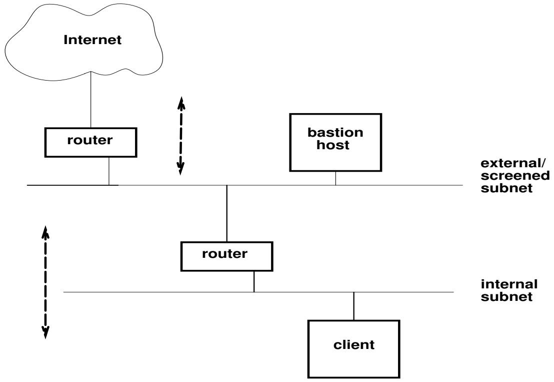
20/21	ftp	53	dns
23	telnet	70	gopher
25	smtp	119	nntp
6000n	X	123	ntp

Common Firewall Design

Consists of two parts which separate the outside network from the internal one:

- Choke (Router)
  - blocks all packets from the outside network destined for the inside network, unless they are destined for the gateway
  - blocks all packets from the inside network destined for the outside, unless they originated from the gateway
- Application Gateway
  - passes data between the two networks

Screened Subnet Example



About Screening Routers

- Inbound and Outbound filtering capability
  - Allows the router itself to appear behind the firewall
  - Some sites want to monitor outbound (ftp) traffic
  - Allows you to recognize address spoofing attempts
  - Convenience of configuration/administration
- Filter all UDP-based services except DNS/bind (port 53) and NTP (port 123)
- Port 2000/6000 are Open Windows and X Windows
- Use different vendors for routers in a DMZ set-up

Firewalls: Gateway Hosts

Implementing the Gateway Host

- Continuing on our present path:
  - Install standard security precautions on the host
  - Then proxy commands to build functionality back into network connection
  - Then fix DNS
  - Finally `sendmail` to hide hosts

Standard Host Gateway Security

- Minimum number of user accounts
- No shared files
- Remove or rename the binaries of all unnecessary commands
- `chmod` all system directories to 711
- Don't run NIS

More Gateway Security

- Turn on system logging
  - Add the tcp_wrappers package available (anon ftp) from cert.org in pub/tools
- Mount all possible disks read only
- Disable all unneeded network services
- The gateway machine cannot play the trusted host game
- Restrict root logins

Secure `ftp` Installation

- If you desire to enable `ftp` for others, do this
- Create /ftp directory with copies of executables and system files; protect them
- Create /etc/ftpusers to restrict accounts which are not allowed to use `ftp`
- Create ./ftp account with disabled password and unwritable home directory
- Create ./bin subdirectory containing ls, unwritable by anyone
- Create ./etc subdirectory with copies of passwd and group files; delete unnecessary lines
- Create ./pub subdirectory, if world writable, anyone can put, otherwise they can just get

Firewalls: Proxy Users

"Proxy" Functionality

- Desire to maintain functionality in a firewall network design
- "Proxy" – the authority to act for another
- Desired functionality:
  - **`ftp`**: file transfer protocol
  - **`telnet`**: user interface to a remote system
  - **`ph`**: access to the on-line phone book
  - **`gopher`**: distributed information delivery system
  - **`DNS`**
  - **`mail`**
  - **`news`**
  - **`X`**

Proxy FTP With Accounts

- Create the ~ftp/pub directory on the gateway machine
- Internal users access the directory via NFS
- Create a special ftpout account on the gateway machine
- Set the shell for this account to /usr/ucb/ftp
- Insure that the ftpout account has a unique UID
- Set the password for ftpout to an asterisk (*)

More Proxy FTP With Accounts

- Make the ftpout account's home directory owned by root, mode 755
- Create ~ftp/.rhosts
  - Owned by root
  - Contains the list of the local users who are authorized to use the ftpout service
- Internal users connect to the gateway and log in as ftpout
  `#` ***`rlogin gateway -l ftpout`***
- Log ftpout account use via syslog

FTP Account Trade Offs

- Doesn't require any new utilities
- `ftp` round-trip connection between internal and external machines is never made
- There is a central repository of documents transferred via ftp
- Users must login
- NFS is used

FTP/Telnet Seamlessly, With Code

- IGateway package available from Sun Microsystems, source available
- Supports `ftp` and `telnet` functionality originating from internal network and directed toward Internet
- Users invoke `iftp` and `itelnet`
  - These programs set up a TCP connection to a daemon on the gateway
  - The daemon interfaces to the outside world, handing packets back and forth

More Proxy Services

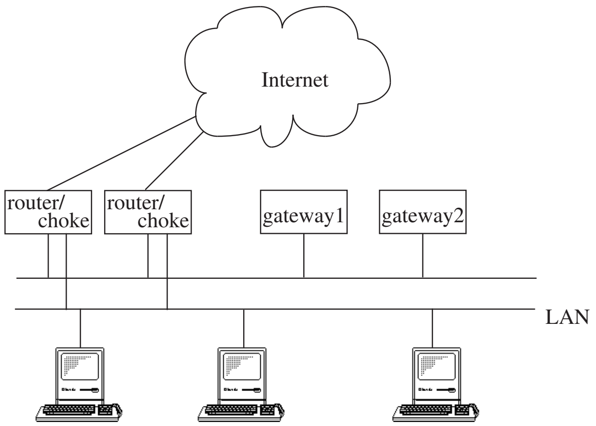
- `socks` is a publically available proxy telnet/ftp server available (anon ftp) from s1.gov
- Modified version of HyperFTP available (anon ftp) from s1.gov
- Patch available for gopher to run with Sun's IGateway software, posted to firewalls@greatcircle.com mailing list
- `xforward` provides a user-level X11 forwarding service, available (anon ftp) from crl.dec.com as /pub/DEC/xforward.tar.Z
- `Tcpr` is a proxy ftp/telnet written in perl, available (anon ftp) from ftp.alantec.com in pub/tcpr

Firewalls: DNS

DNS' Firewall Role

- Help hide internal network structure
- Replace functionality through mail gateway interaction

our.domain Master DNS Servers



BIND hosts Data File

- Use MX records to accept mail on behalf of the domain, e.g., our.domain
- The location of this file is specified in the named.boot file
- Sample:

```
      Configuration information for our.domain
@   IN   SOA  gw1.our.domain. postmaster.gw1.our.domain. (
                        296     ; Serial
                        3600    ; Refresh
                        300     ; Retry
                        86400   ; Expire
                        3600 )  ; Minimum
    IN   NS   gw2.our.domain.
    IN   NS   gw1.our.domain.
    IN   MX   0 gw2.our.domain.
    IN   MX   10 gw1.our.domain.
    IN   A    128.115.32.7        ;for old mailers
```

Additional DNS Possibilities

- Some sites don't want internal machine information available
- With MX records and properly configured supporting applications, internal machine information is not needed

DNS Possibilities (cont.)

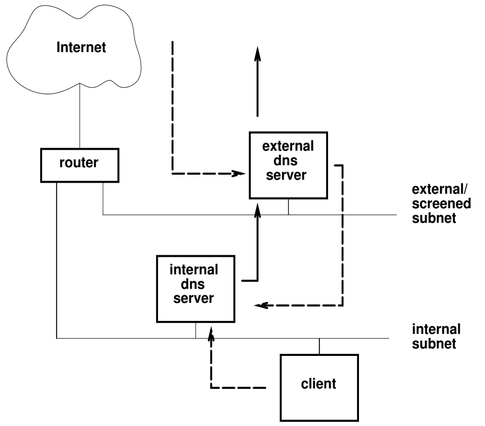
- Can run 2 sets of domain name servers for the site, internal and external servers
  - External servers keep data for just the machines which are reachable from the outside world
  - External servers' resolv.conf points to the internal servers
  - Internal servers contain all the information for the internal nets
  - Internal servers forward request for outside data to the external servers, which proxies the question

DNS Proxy/Forwarding

- If you designate the external DNS servers as site forwarders, all the off-site queries are sent to the forwarders first
- You must restrict the internal name server further, stopping them from trying to contact an off-site server, by designating it as a slave server
- Add the following lines in the named.boot file

```
forwarders 128.115.32.1 128.115.32.2
slave
```


Dual-DNS Example



Firewalls: Mail

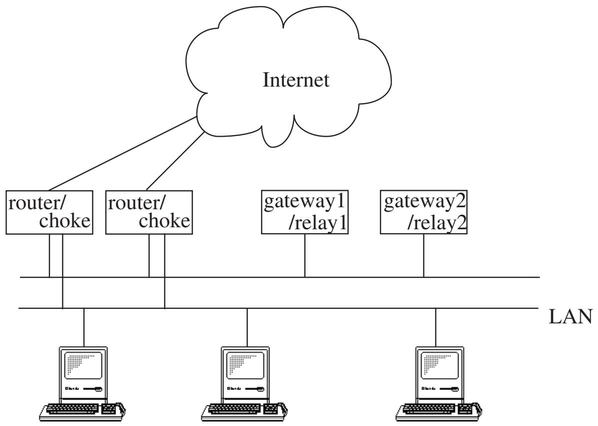
Mail Gateways

- Gateway sometimes describes software that performs specific conversions at layers above the network layer
- Mail gateways convert electronic mail from one format to another
- Our gateway is an application level electronic mail gateway that:
  - Routes mail between the internal LAN and the Internet and then
  - Rewrites outbound mail headers appropriately

Where We Are Going

- First we'll discuss site-wide mail topology
  - Proposed scheme
  - What the benefits are
- Then we'll cover how to modify the sendmail.cf file to support our mail topology

Firewall Mail Topology



More Firewall Mail Topology

- Relay: A machine that will relay mail to the Internet on behalf of a standalone, fileserver, or client
- Fileserver: An NFS fileserver that contains /usr/spool/mail for CLIENTs
- Client: Users' machines that do not contain a local /usr/spool/mail

Security Benefits of Electronic Mail Design

- Hide the local structure of internal organization by rewriting, thus:

  `user@machine.our.domain`

  becomes:

  `user@gateway.our.domain`

  or

  `user@our.domain`

Administrative Benefits to Mail Design Philosophy

- The configuration file on the master mail machine(s) must be able to handle connections to external networks
- The rest of the machines on site can have a simple configuration file that forwards mail to the master(s)
- Built in redundancy gives you robustness and load balancing (costs more, but avoids network bottleneck and single point of failure)
- With header rewriting, users' email addresses never have to be relearned or re-distributed
- The master machine(s) can be replaced and the site email addresses can remain unchanged

More Mail Design

- Source machine and destination address will determine what mailer and what machine will get involved in the mail delivery

Source On	Mailer	Delivered To	Destination
Relay	tcp	Internet Host	Internet
	localtcp	FS/SA	Localnet
Standalone	tcp	Relay	Internet
	localtcp	FS/SA	Localnet
	local	SA	Local
FS	tcp	Relay	Internet
	localtcp	FS/SA	Localnet
	local	FS	Local
Client	tcp	Relay	Internet
	nfs	FS/SA	Localnet

Designing a sendmail.cf For Your Site

- **Don't start from scratch - modify an existing sendmail.cf**
- **Design the sendmail.cf to support your site's mail topology**
- **Don't panic; you can do this!**

Design Goals

- **Deliver all local mail to fileservers only**
- **Rewrite all headers to appear as though they originated from fileservers rather than clients – reply headers point to servers**
- **Route all outbound mail through mail gateways**
- **Hide internal LAN structure by rewriting all outbound mail headers to appear as if the mail originated from the gateways**
- **Accept all inbound mail on GATEWAYs**
- **Load balance mail traffic through mail GATEWAYs**

Defining a FILESERVER class

```
# if defined(SERVER)
DF/**/SERVER
```

- Defining a macro for later use
  `DJ$?F$F$|$j$.`
- Ruleset 0
  `R$+     $?F$#nfs$@$F$:$1$|$#local$:$1$.`

Fileserver Header Rewriting

```
Mlocal,P=/bin/mail, F=rlsSDFMmn,
       S=10, R=20, A=mail -d $u
Mnfs,  P=[IPC],  F=msDFMuXLI,
       S=11, R=20, A=IPC $h, E=
s10
R@   MAILER-DAEMON  errors to mailer-daemon
S20
S11
R$-  $@$1@$J
```

Route Outbound Mail Through Gateways

- Defining a Macro
  `DR/**/RELAY`

- Ruleset 0
  `R$*<@$+>$*        $#tcp$@$R$:$1<@$2>$3`

Rewrite Outbound Addresses

```
Mtcp, P=[IPC],  F=MSDFMUEXL, S=15,
        R=14, A=IPC $H, E=R

S14
R$- R<@$+>    $@<@$1>$2       resolve <ROUTE-ADDR>
R$*<@$+.UUCP $:$2!$1<@$J>    put local name on UUCP
R$*:$*        $1.$2           map colons to dots

S15
R$-           $@$?D$1@$D$|$1@$J$. local sender
R$*           $:$>14$1            normal rewriting
R$*<@$=L>     $:$?D$1@$D|$1@$2$.  a@ourhost => a@ourdomain
R$*<@$=L.$D>  $:$?D$1@$D|$1@$2$.  a@ourhost.ourdomain=>
                                          a@ourdomain
```


Accept Inbound Mail To Domain Name

- Remember the BIND MX Record!

- Ruleset 0
  `R$*<@$D>$*  $#local$:$1  user@ourdomain`

- Maintain a global "aliases" file

Aliases

- For forwarding mail from the gateway to the internal mail server machines

- For forwarding between internal mail server machines

- The `newaliases` command is `sendmail -bi` and must be run when aliases file changes

- Distribute it with `rdist` to all mail FILESERVER machines

Using `rdist` To Keep Aliases In Sync

```
# Distribution file for rdist
HOSTS = ( gateway2 fileserver1 fileserver2 )
FILES = ( /usr/lib/aliases )
${FILES} -> ${HOSTS}
        install -h ;
        special "newaliases" ;
```


Using a Makefile To Create Your Site's sendmail.cf Files

```
#!/bin/make -f
#  Create configuration files for our.domain hosts
#  GATEWAY:  machine directly on  Internet
#  FILESERVER:  NFS fileserver with
#       /usr/spool/mail for CLIENTS.
#  RELAY:  machine that relays mail to Internet
SRC  =     sendmailcf.c
CPP  =     cc -E $(SRC)
all:  relay1.cf  relay2.cf standalone.cf \
      fileserverA.cf fileserverB.cf fileserverC.cf \
      client_of_A.cf  client_of_B.cf  client_of_C.cf
#RELAY MACHINES
relay1.cf relay2.cf:  sendmailcf.c
        $(CPP) -DGATEWAY  > $@
#FILESERVER MACHINES
fileserverA.cf: sendmailcf.c
        $(CPP)  -DRELAY1.our.domain > $@
fileserverB.cf fileserverC.cf: sendmailcf.c
        $(CPP)  -DRELAY2.our.domain > $@
```


Rest of Makefile

```
#CLIENT MACHINES
client_of_A.cf: sendmailcf.c
        $(CPP) -DFILESERVER=fileserverA.our.domain \
                -DRELAY=relay1.our.domain > $@
client_of_B.cf: sendmailcf.c
        $(CPP) -DFILESERVER=fileserverB.our.domain \
                 -DRELAY=relay2.our.domain > $@
client_of_C.cf: sendmailcf.c
        $(CPP) -DFILESERVER=fileserverC.our.domain \
                 -DRELAY=relay2.our.domain > $@
#STANDALONE MACHINES
standalone.cf:  sendmailcf.c
        $(CPP) -DRELAY=relay1.our.domain > $@
```

Ethics

RBK – ethics.00 Jul 29, 1993

ETHICS

- Growth in computer use has uncovered new problems
  - More data on line
  - More SENSITIVE data on line
- Connection of more computer networks likewise
- "The Worm" has heightened sensitivity
- Computer ethics is a new discipline without a long history
- We need to know how to address ethics, privacy, and security problems so that we protect rights and still get work done
- Problems are new – no extant body of wisdom

RBK – ethics.01 Jul 29, 1993

Reading Others' Files

- All systems administrators must earn the users' TRUST; without it, all security/privacy issues are moot
- Privacy of electronic mail: When should you read it? How?
  - Mail system repair
  - `grep` for mail headers
- Reading random "open files"
  - Often a bad plan
  - "Unlocked door" does not mean stealing is OK
  - → Permission of someone else is important
  - Compress and encrypt private files which must be kept online
- Users need to learn the above – IN ADVANCE

RBK – ethics.02 Jul 29, 1993

Violators

- What should we do with first-time violators?
  - Have a gentle talk with them
  - Have a less-than-gentle talk with them
  - Slap their wrist
  - Demote them
  - Dock their pay
  - Transfer them
  - Fire them
- What should we do with repeat offenders?
- [This slide includes a story about a new employee and the results of his action]

RBK – ethics.03 Jul 29, 1993

Ethical Problems of the Mail System

- Mailing private mail to the wrong alias, e.g.:
  % mail george Please thank everyone for me ...
- Electronic mail does not seem to have the permanence of written, yet is just as solid when printed
- Discussing private issues in electronic mail is quite vulnerable to mishaps
- Deliberate impersonation is not hard enough
  - Data switch accidents
  - Modems
  - `telnet` hostname 25
  - Unattended terminals
  - X terminals not fully logged out
- Sarcasm doesn't work through E-Mail, by the way

Use and Abuse of Superuser Access

- Non-superusers all want to be superusers
- Reading others' files is an invitation to trouble and distrust
- "Operator wars" in almost any environment seems to cause incredible, far-reaching problems
- Log all superuser access – you never know when you'll be tracking an intruder
- Use `sudo` which limits superuser access
- Change 'root' to 'clerk'
- All superusers get beepers
- "License to hack" can help when users wish to experiment

Accounting

- If you get money for your services, make sure your accounting is first-rate
- If you don't have first rate accounting, the auditors will remind you and you'll look bad
- Don't ever circumvent accounting procedures – or they'll ALL be suspect
- Every user gets one signon (no borrowing or sharing) – otherwise you'll be unable to track problems when they occur
- Use consistent uids and gids across a distributed system

GNU's Software and the Copyleft

- GNU (aka The Free Software Foundation) distributes a wealth of marvelous, free software
- GNU's software is all protected by the "Copyleft"
- The GNU organization is a bit zealous ("religious")
- The "Copyleft" says that if you use any of their software in your product, it is Copylefted, too
- Some organizations therefore refuse to import any GNU software

Detecting Problems

- Watch for new accounts and new files (see Tripwire)
- Watch for modified programs/data (see Tripwire)
- Find causes of poor performance or service denial
- Make sure accounting makes sense

Documentation

- Document all system events
- Document all actions you take (with time/dat)
- Document all telephone & e-mail conversations, including names, dat/time, conversation contents
- Log books really help (offline, of course)
  - Photocopy new logbook pages each day
  - Sign and date them
  - Submit to document custodian for secure storage

Potential Entities to Contact

- Notify systems personnel
- Maybe notify response times
- Maybe notify your organization's legal counsel
- Maybe notify law enforcement personnel
  - Contact them soon, not when things are broken
  - Build a relationship
  - Teach them what you can and can not do
  - Give them a tour
- Notifying press brings its own problems (see press section)
  - If you must do it, follow those rules
  - Use your company's PR office

So Someone Has Deliberately Invaded Your System

- Don't Panic
- Stop the bleeding
- Assess the damage
- Fix the problem
  - Plug the leak
  - Contact CERT if necessary (412-268-7090 24 hours/day or cert@sei.cmu.edu) – keep this info offline
- Details Follow

Planning Ahead to Succeed in a Security Emergency

- Don't panic
- Don't hire people who will panic
- Decide whether to "pull the plug"
  - What is your risk?
  - What is your net neighbors' risk?
- Who will make all these decisions?
  - Better have a written list ahead of time!
  - Better know how to decide who's in charge when some are missing
  - Don't let 'pride' or misguided aphorisms dictate policy

Handling the Media in an Emergency

- In a word: Don't
- Tell your people not to speak with the media
- Designate a spokesperson
- Never release technical details in real time
- IETF has done a computer security handbook; use ftp to `sei.cmu.edu` and find it in `pub/ssphwg/rfc1244.txt`

Who's In Charge During the Emergency?

- Administration wants to be in charge – it's their job!
- That doesn't mean they're right
- Sometimes the fire-fighters should be in charge
  - Higher efficiency
  - More results
- Whoever is in charge – make sure everyone knows and agrees before the problem arises. You need an absolute dictator in charge

Investigating Breakins

- First defense is the `last` command
- Second defense is command accounting
  - Save the last week or so of command accounting
  - Get a version of `sa` that allows you to examine it easily when needed
- Only extreme cases may require "keyboard monitoring" – and now there's legal problems with that!

Monitoring

- Use **`netstat -a`** to see all active network connections
- Use **`netstat -r`** to see network routes – watch for numbers instead of names (they indicate bogus routes)
- You probably already log **`ftp`** connections (use **`ftpd -l`** in **`/etc/inetd.conf`**)

Ethics on NetNews

```
/* Written  9:33 am  Aug 21, 1989 by 2014_5001@uwovax in
                              prisma:soc.motss */

                 "See ya all"

All ye homos, see you at my admissions office.
After your first HIV infection it oughtn't be long.

----------------------------------------------------
|  Satan @ uwovax.bitnet | See you in hell, buster |
----------------------------------------------------
|  Don't believe what people say about hell:  the  |
|  flames are nice and warm.  I love flames. |------
|--------------------------------------------|

Note:  use uwovax.bitnet as a fast gateway to ABYSS.HELL.

/* End of text from prisma:soc.motss */
```

- Mr. 'Satan' wrote over a dozen of these style notes all over netnews on Rob's birthday

What To Do With NetNews Abusers?

- Obligation to employers
- Obligation to other users
- Obligation to humanity
- Obligation to the first amendment
- Options
  - Discussion
  - Wrist-slap
  - Loss of posting rights
- It may not matter, of course