

# Unix System Administration

**Chris Schenk**

**Lecture 18 – Tuesday Mar 20**

**CSCI 4113, Spring 2007**

# Apache Resources

- Google: rewrite http to https
  - RewriteEngine On, RewriteCond %{HTTPS} != On
  - RewriteRule – Uses regular expressions
    - Match things AFTER the hostname: “^/internal(.\*)\$”
    - Rewrite the ENTIRE url: “https://coolname....\$1 [R,L]”
- “Directory” directive, read the rule ordering!
  - Order allow,deny -- deny,allow
  - Last-match-wins ordering
- Lab updated with SSL info from Nate Watkins
  - Describes certificate verification tree

# Digital Signatures

- We keep talking about 'signing' certificates
  - How do we actually accomplish this?
- Signatures require a public/private keypair
  - We can encrypt a piece of data with our private key
  - No one else has our private key, difficult to forge ciphertext
- But private/public key algorithms are block ciphers
  - Means fixed input gives the same fixed size output
- We don't want our signature to be as long as the message

# Cryptographic Hashes

- $h = h^{-1} = h(m)$
- A 'hash' by itself takes a variable length input
  - Always produces a fixed size output
  - MD5, SHA1, CRC, and others
- Cryptographic hashes are functions that add certain security properties suitable for some information security
  - Verifying file integrity
  - Signing email to demonstrate originality
  - Password hashes

# Cryptographic Hash Properties

- Cryptographic hashes have three general properties
- Preimage resistance
  - Given a hash value  $h$ , it is difficult to find the original message
- Second preimage resistance
  - Given a hash value  $h$  and its message  $m_1$ , it is difficult to find another input  $m_2$  such that  $h = h(m_1) = h(m_2)$
- Collision resistance
  - It should be difficult to find two messages  $m_1$  and  $m_2$  such that  $h(m_1) = h(m_2)$

# Cryptographic Hashes and Signatures

- Digital Signatures follow a Hash-Then-Sign paradigm
- Remember, a block cipher produces output the same size as its input
  - We want to sign our data, but we want to keep the sig. small
  - In fact, it'd be better to keep it a fixed size!
- So let's first hash our data **BEFORE** signing
  - Takes the input to the block cipher to a fixed size
    - Size depends on the hash used (MD5, SHA1, etc)
- Why don't we hash **AFTER** using block cipher?

# Where Crypto Hashes Are Used

- Passwords
  - /etc/shadow (under linux)
- Digital signatures
  - PGP email identification
  - File verification – debsums
  - SSL certificate requests and signed certificates
  - SSH public keys for quick(er) key identification
  - Idea publications
    - Write a paper on a topic and hash the paper
    - Publish the hash in the local newspaper

# How to Verify Hashes

- You need a published hash
  - Provided by people distributing file
- Download the file and the hashes
  - 'md5sum' verifies md5 hashes
  - PGP – pretty good privacy signatures
    - Better than simple md5sums
- Like SSL certificates, this requires effort
  - But is what is necessary for security

# MD5 – A Broken Cryptographic Hash?

- **Let's take a look at the wikipedia post:**
  - Because MD5 makes only one pass over the data, if two prefixes with the same hash can be constructed, a common suffix can be added to both to make the collision more reasonable. And because the current collision-finding techniques allow the preceding hash state to be specified arbitrarily, a collision can be found for any desired prefix -- for any given string of characters X, two colliding files can be determined which both begin with X. All that is required to generate two colliding files is a template file, with a 128-byte block of data aligned on a 64-byte boundary, that can be changed freely by the collision-finding algorithm.
- **Byte boundaries and template files are mentioned**

# MD5 – A Broken Cryptographic Hash?

- Let's look at this a little further
  - <http://www.cits.rub.de/MD5Collisions/>
  - Site has an example of a hash collision for two documents
- A 'prefix' was mentioned on the previous slide
  - This prefix can be modified while the rest of the document is left the same
- What type of attack is this?
  - Think in terms of the cryptographic hash properties from earlier
- Does this make MD5 bad to use for hashes?

# Message Authentication Codes

- MACs
  - Not to be confused with network Layer 1 addresses!
- MACs use hash functions to tag messages
  - Used by our RNDC utility for BIND
  - All messages are authenticated with a 'tag'
- Not the same as a digital signature!
  - Digital sigs use assymmetric keys
  - MACs use a symmetric key with the hash
- Need to be resistant to chosen plain-text attacks

# Email

- One of the primary uses of the internet
  - Aside from porn
- Mail routing almost as complicated as IP
  - Enter MX records from DNS
- Email system consists of parts called 'agents'
  - Each agent is responsible for a part of the system
- Five different agents
  - user, submission, transfer, delivery, access
- Mail servers perform two roles (usually)
  - submission and transfer agents

# Email and DNS

- Email closely tied to DNS information
  - Many rules for mail routing based on DNS info
  - MX records, reverse PTR records, A records, etc
- MX records are the routing table for email
  - Like layer two's information
    - `netstat -nr`
  - Maintained by the admin of the domain
- Can restrict mail based on PTR records
  - IF A record != PTR record, don't route mail
    - For incoming hosts!

# DNS MX records

- Usually an RR has 5 pieces of information
  - Node in tree, TTL, class, type, value
- Value can have more than one piece of info
  - SOA has 7 pieces
    - Refresh, retry, etc
  - MX has two
- MX contains the priority of the mail exchanger
  - The LOWER the number, the HIGHER the priority
  - IN MX 50 cs.colorado.edu.
  - IN MX 10 coolname.cs.colorado.edu.

# Email – User and Access Agents

- Mail User Agent (MUA) – end-user programs
  - Pine, webmail, thunderbird, etc
- May have Mail Access Agent (MAA) as well
  - Allows reading mail via POP or IMAP
- Might have MIME rendering/encoding functionality
  - Converting binary files to base64 and back
- Should allow access to mail header fields
  - Sometimes it's hard to find them, but they are critical to debugging

# Email – Submission and Delivery Agents

- Mail Submission Agent (MSA)
  - Receives mail from MUAs on port 587
  - Also useful for security and validity of email
    - Makes sure the right headers are intact and valid
    - Potentially scan for viruses
- Mail Delivery Agent (MDA)
  - Actually sends a piece of mail to a user's inbox
    - Once an MTA receives a piece of mail destined for the local machine
    - Can be any program that accepts STDIN
  - Most common is `procmail`

# Email – Transfer Agent (MTA)

- Delivers mail between machines
  - May queue of mail in local files for future delivery
- Most machines communicate on port 25 for delivery
  - Now also listen on port 587 like MSAs
- Sending mail to another machine is called 'relaying'
  - 'open' relays send mail to anyone (evil)
- MTAs speak SMTP on port 25
  - We'll look at an example transaction

# Anatomy of an Email Message

- Three main parts to an email message
  - envelope, headers, body
- Envelope
  - You never see this, it's used internally by the MTA
  - (Usually) corresponds to the 'From' and 'To' headers
    - But not always!
- Body
  - Actual content of the message
  - MUST be in plain text, all binary data is converted to base64
  - The body is separated from headers by a blank line

# Anatomy of an Email Message (cont)

- Headers
  - These are Key: Value pairs that conform to RFC822 standards
  - Some are mandatory: From, To, and others
    - If omitted, usually considered spam
  - Others are optional: Subject, Cc, Reply-To
  - Still others are ignored by the mail system
    - All headers are propagated, whether read or not
  - Headers that start with 'X-' are application specific
    - But again, are still sent even if unused

# Reading Email Headers

- Important to know for tracing emails to their source
  - and diagnosing other problems
- The `Received` lines are the most important
  - Each intervening MTA adds a `Received` header
  - Details the route taken by that piece of email
- `Received: from refuge.Colorado.EDU (refuge.colorado.edu [128.138.238.167])  
by mx13.Colorado.EDU (Spam Firewall) with ESMTTP id 8DC451830CE  
for <schenkc@ucsub.Colorado.EDU>; Tue, 4 Apr 2006 11:36:48 -0600 (MDT)`  
`Received: from refuge.Colorado.EDU (localhost [127.0.0.1])  
by refuge.Colorado.EDU (8.13.6/8.13.3/UnixOps+Hesiod+SSL) with ESMTTP id  
k34HamxB000970  
for <schenkc@colorado.edu>; Tue, 4 Apr 2006 11:36:48 -0600 (MDT)`

# Other Header Lines

- Message-Id – Added by the originating MTA, unique
- Return-Path – Supposed to contain the *Envelope* sender's address, bounced mail is sent here
- Date – Added by the originating MUA
- Received – Mixed in with the above
- From, To, Subject, etc – All follow the above
- Many other fields (usually starting with X-)

# MIME Encoding

- Makes use of header fields

- ```
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary="N00000D0151102.DB1020"
Content-Disposition: inline
```

- Content-Type indicated – there are many types
  - 'multipart' indicates there will be several documents
  - 'mixed' means they may be different types
- boundary – A string inserted between each attachment
  - So the MUA knows how to split the documents

# Email Addresses

- Complexity of mail systems stems from different addressing schemes
  - UUCP addressing: ajax!boulder!mroe!target
    - Used to have to know the hops between you and destination
  - Modern addressing – lots of variation here
    - Chris Schenk <schenkc@colorado.edu>
    - “Joe Blow” <jb@blowme.net>
    - Hans.and.Franz@pumpUup.net
    - username@machine.com
  - Mixed Addressing - ucsb!schenkc@colorado.edu

# Aliases

- Provide for the re-routing of email
  - From one to one or many different destinations
- The addresses in To, Cc and From fields may be modified
- Aliases are recursive
  - With limitations to avoid mail loops (usually large though)
- Two general locations for aliases
  - /etc/aliases
  - /etc/mail/aliases

# Classic Sendmail Alias Files

- `/etc/aliases` is a flat file database
  - Converted to a hash database for efficient lookups
  - The 'newaliases' (historical) command creates the hash files
  - Other types of files use hashes as well
- **Basic format:** `key: value[,value2[,value3[,...etc]]]`
  - Comments (lines starting with `#`) and blank lines are ignored
  - Simple alias is a redirect to another user: `root: schenkc`
  - Redirect to another host: `root: schenkc@colorado.edu`
  - Alias a list of users: `csci4113: chris,phil,bob,jane`

# Aliases File (cont)

- Advanced format
- **Include files:** `graders: :include:/etc/postfix/graders`
- **Append to a file:** `webmaster: "/dev/null"`
  - The quotes preserve the slashes and path is always absolute
- **May also invoke a command:**
  - `schenkc: "|/usr/bin/auto-responder"`
  - This pipes the mail to stdin of the listed command
- **Mailing lists use piping to manage list email**
  - `csci4113: "|/usr/local/majordomo/bin/wrapper -L -l csci4113"`

# Postfix Alias Files

- Have a different syntax than traditional files
  - `key value[,value2]` (notice the space)
- Keys and values are many times fully qualified
  - `schenc@c@elijah.cs.colorado.edu schenc@c@colorado.edu`
- Aliases are looked up in files by order of appearance
- `alias_maps: hash:/etc/aliases, hash:/etc/extra_aliases`
- When mixing alias files, Postfix files take precedence
  - Sometimes, in terms of matching lookups – it's odd behavior

# .forward Files

- Users can forward mail to another location
  - .forward behaves similarly to alias files
- File exists in user's home directory
  - Permissions must be only readable and writable by owner!
- Can cause mail loops if not carefully managed
  - Internal alias loops are handled by servers, not forwarding loops
- **Syntax:** `\schenk , " | /usr/bin/vacation -f vacation.msg"`
  - The preceding `\` says force delivery of mail without forwarding