Case Study: Access control

Case Studies in Access Control

- Joint software development
- Mail

Joint Software Development

Situations

- Small team on a single machine
- Medium-to-large team on a LAN
- Large, distributed team, spread among several organizations

Case Studies in Access Control
Joint Software Development
Situations
Roles
Permissions
Why Enforce Access Controls?
Classic Unix Setup
ACL Setup
Reviewer/Tester Access
Medium-Size Group
Basic Structure
Version Control Systems
Why use a VCS?
Note Well
Structure of a VCS
Permission Structure
They're Not SetUID!
The Repository
Large Organization
Complications
Mailers

Roles

Case Studies in Access Control
Joint Software Development
Situations
Roles
Permissions
Why Enforce Access Controls?
Classic Unix Setup
ACL Setup
Reviewer/Tester Access
Medium-Size Group
Basic Structure
Version Control Systems
Why use a VCS?
Note Well
Structure of a VCS
Permission Structure
They're Not SetUID!
The Repository
Large Organization
Complications
Mailers

- Developer (i.e., can commit changes)
- Tester
- Code reviewer

Permissions

Case Studies in Access Control

Joint Software Development

Situations

Roles

Permissions

Why Enforce Access Controls?

Classic Unix Setup

ACL Setup

Reviewer/Tester Access

Medium-Size Group

Basic Structure

Version Control Systems

Why use a VCS?

Note Well

Structure of a VCS

Permission Structure

They're Not SetUID!

The Repository

Large Organization

Complications

Mailers

- We want the technical mechanisms to reflect the organizational roles
- The real challenge: mapping the organizational structure to OS primitives
- Why?

Why Enforce Access Controls?

Case Studies in Access Control
Joint Software Development
Situations
Roles
Permissions
Why Enforce Access Controls?
Classic Unix Setup
ACL Setup
Reviewer/Tester Access
Medium-Size Group
Basic Structure
Version Control Systems
Why use a VCS?
Note Well
Structure of a VCS
Permission Structure
They're Not SetUID!
The Repository
Large Organization
Complications
Mailers

- Protect software from outsiders reading/stealing it
- Protect against unauthorized changes

⇒ That can include internal rivalries

- Know *who* made certain changes?

Classic Unix Setup

- Assume you just have user/group/other permissions, with no ACLs. How would you set things up?
- Put all developers in a certain group
- Make files and directories group readable/writable
- Decision to turn off "other" read access is site-dependent

ACL Setup

- Could add each developer individually
- Bad idea — if a developer leaves or joins the group, many ACLs must be updated
- Still want to use groups; vary group membership instead
- Advantage: can have multiple sets of group permissions — why?

Reviewer/Tester Access

- Reviewers and testers need read access
- They do not need write access
- No good, built-in solution on classic Unix
- With ACLs, one group can have `r/w` permissions; another can have `r` permissions

Medium-Size Group

- No longer on single machine with simple file permissions
- More need for change-tracking
- More formal organizational structure

Basic Structure

Case Studies in Access Control
Joint Software Development
Situations
Roles
Permissions
Why Enforce Access Controls?
Classic Unix Setup
ACL Setup
Reviewer/Tester Access
Medium-Size Group
Basic Structure
Version Control Systems
Why use a VCS?
Note Well
Structure of a VCS
Permission Structure
They're Not SetUID!
The Repository
Large Organization
Complications
Mailers

- Basic permission structure should be the same
- Again: use group permissions as the fundamental permission unit
- Limits of non-ACL systems become more critical

Version Control Systems

Case Studies in Access Control
Joint Software Development
Situations
Roles
Permissions
Why Enforce Access Controls?
Classic Unix Setup
ACL Setup
Reviewer/Tester Access
Medium-Size Group
Basic Structure
Version Control Systems
Why use a VCS?
Note Well
Structure of a VCS
Permission Structure
They're Not SetUID!
The Repository
Large Organization
Complications
Mailers

- For medium-size projects, use of a version control system (i.e., CVS, Subversion, Mercurial, RCS, etc.) is *mandatory*
- (Why?)
- What are the permission implications of a version control system?

Why use a VCS?

Case Studies in Access Control
Joint Software Development
Situations
Roles
Permissions
Why Enforce Access Controls?
Classic Unix Setup
ACL Setup
Reviewer/Tester Access
Medium-Size Group
Basic Structure
Version Control Systems
Why use a VCS?
Note Well
Structure of a VCS
Permission Structure
They're Not SetUID!
The Repository
Large Organization
Complications
Mailers

- Auditability — who made which change?
- When was a given change made?
- Can you roll back to a known-clean version of the codebase?
- What patches have been applied to which versions of the system?

Note Well

- All of those features are important just for manageability
- Security needs are strictly greater — we have to deal with active malfeasance as well as ordinary bugs and failures

Structure of a VCS

Repository Master copy; records all changes, versions, etc.

Working copies Zero or more working copies. Developers *check out* a version from the repository, make changes, and *commit* the changes

Permission Structure

Here are the Unix client commands for RCS, CVS, Mercurial, and Subversion. What are the security implications?

```
$ ls -l /usr/bin/ci /usr/bin/cvs  \
    /usr/pkg/bin/hg /usr/pkg/bin/svn
-r-xr-xr-x  1 root  wheel  /usr/bin/ci
-r-xr-xr-x  1 root  wheel  /usr/bin/cvs
-rwxr-xr-x  1 root  wheel  /usr/pkg/bin/hg
-rwxr-xr-x  1 root  wheel  /usr/pkg/bin/svn
```

They’re Not SetUID!

Case Studies in Access Control
Joint Software Development
Situations
Roles
Permissions
Why Enforce Access Controls?
Classic Unix Setup
ACL Setup
Reviewer/Tester Access
Medium-Size Group
Basic Structure
Version Control Systems
Why use a VCS?
Note Well
Structure of a VCS
Permission Structure
They’re Not SetUID!
The Repository
Large Organization
Complications
Mailers

- They execute with the permissions of the invoker
- They could try to do access control, but it’s meaningless — anyone else could write code to do the same things
- The permission structure of the repository is what’s important

The Repository

- Essential feature: developers must have write permission on the directories
- File permissions are irrelevant; old files can be renamed and unlinked instead of being overwritten
- (Potential for annoyance if new directories are created with the wrong permission — must set `umask` properly)
- But — what prevents a developer with write permission on the respository from doing nasty things?
- Nothing. . .

Large Organization

Case Studies in Access Control
Joint Software Development
Situations
Roles
Permissions
Why Enforce Access Controls?
Classic Unix Setup
ACL Setup
Reviewer/Tester Access
Medium-Size Group
Basic Structure
Version Control Systems
Why use a VCS?
Note Well
Structure of a VCS
Permission Structure
They're Not SetUID!
The Repository
Large Organization
Complications
Mailers

- Use client/server model for repository access
- Most users (including developers) have no direct access to the VCS repository
- Either build access control into VCS server or layer on top of underlying OS permissions
- But — must restrict what commands can be executed on repository by developers

Complications

Case Studies in Access Control
Joint Software Development
Situations
Roles
Permissions
Why Enforce Access Controls?
Classic Unix Setup
ACL Setup
Reviewer/Tester Access
Medium-Size Group
Basic Structure
Version Control Systems
Why use a VCS?
Note Well
Structure of a VCS
Permission Structure
They're Not SetUID!
The Repository
Large Organization
Complications
Mailers

- If you rely on OS permissions, *something* has to have `root` privileges, to let the repository part of the process run as that user
- If the VCS itself has a root component, is it trustable?
- If you use, say, `ssh`, is the command restriction mechanism trustable?
- If you rely on VCS permissions, you need to implement a reliable authentication and ACL mechanism
- All of this is possible — but is it *secure*?

Mailers

Mailers

- Issue of interest: local mail delivery and retrieval
- Surprisingly enough, network email doesn't add (too much) security complexity

Issues

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

- Email *must* be reliable
- Users must be able to send email to any other users
- The system should reliably identify the sender of each message
- All emails should be logged
- Locking is often necessary to prevent race conditions when reading and writing a mailbox
- Authentication

Accepting Mail

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

- Must accept mail from users
- Copy it, either to protected spool directory for network delivery or directly to recipient's mailbox

Spool Directory

- If the mailer is setuid, it can copy the email to a protected directory with no trouble
- If the directory is world-writable but not world-readable, you don't even need setuid — add a random component to the filenames to prevent overwriting
- (Homework submission script does this)
- File owner is automatically set correctly, for use in generating `From:` line

However. . .

- Cannot securely write metadata for such directories — others could overwrite the metadata file
- Cannot prevent users from overwriting their own pending email
- Listing the mail queue still requires privilege

Local Access or Client/Server?

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

- For client/server, issues are similar to VCS: authentication, root programs, restricting actions, etc
- For local access, must confront permission issues
- This is complicated by the many different versions of Unix over the years

Client/Server

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

- Standardized, (relatively) simple access protocols, POP and IMAP
- For ISP or large enterprise, neither need nor want general shell-type access to mail server
- Large system mailers have their own authentication database
- Does not rely on OS permissions
- But — a mail server bug exposes the entire mail repository
- Also — how do users change their passwords?

Bug Containment

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

- Separate programs into two sections:
  - Small, simple section that does authentication and changes uid (must run as `root`)
  - Large section that runs as that user
- Major advantage: security holes in large section don't matter, since it has no special privileges
- Much more on program structure later in the semester

Local Mail Storage

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

- Where is mail stored? Central mailbox directory or user's home directory?

- Note that mail delivery program must be able to (a) create, and (b) write to mailboxes

- If mailbox is in the user's directory, mail delivery program must have root permissions

Central Mail Directory

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

- We can put all mailboxes in, say, `/var/mail`
- What are the permissions on it?
- If it's writable by group `mail`, delivery daemon can create new mailboxes
- Make mailboxes writable by group `mail`, and owned by the recipient?
- Permits non-`root` delivery — but how do new mailboxes get created *and* owned by the user?

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

Dangers of User-Writable Mailbox Directories

Permission escalation	`ln -s /etc/passwd /var/mail/me`
Vandalism	`rm /var/mail/you`
Denial of service	`touch /var/mail/does-not-exist-yet`

Defending Against These Attacks

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

Escalation Check mailbox permissions and ownership before writing (note: watch for race conditions)

Vandalism Set "sticky bit" on directory

DoS Remove (or change ownership of) mailboxes with wrong ownership

Note well: most of these are trickier than they seem

Delivering Mail to a Program

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

- Most mail systems permit delivery of email to a program
- Must execute that program as the appropriate user
- (Who is the “appropriate” user? Note that on Solaris, you may (depending on system configuration) be able to give away files)
- Implies the need for `root` privileges by the local delivery program

Privileged Programs

- What must be privileged?
- What privileges?
- Local delivery needs some privileges, frequently `root`
- Delivery to a program always requires `root`
- The mail reader?

Privileged Mail Readers

Case Studies in Access Control
Joint Software Development
Mailers
Mailers
Issues
Accepting Mail
Spool Directory
However. . .
Local Access or Client/Server?
Client/Server
Bug Containment
Local Mail Storage
Central Mail Directory
Dangers of User-Writable Mailbox Directories
Defending Against These Attacks
Delivering Mail to a Program
Privileged Programs
Privileged Mail Readers
Many More Subtleties
Why is it Hard?

- The System V mail reader was `setgid` to group `mail`
- Could delete empty mailboxes
- More importantly, could create lock files by linking in the mailbox directory
- But — note the danger if the mailer was buggy
  "You don't give privileges to a whale" (about 21K lines of code. . .)

Many More Subtleties

- Writing a mailer is *hard*
- I’ve barely scratched the surface of the design decisions, even the permission-related ones
- Complicated by varying system semantics

Why is it Hard?

- Mailers cross protection boundaries
- That is, they copy data from one permission context to another
- Both can be arbitrary userids
- Simply importing data to a userid is a lot easier
- In addition, a lot of functionality is needed
- Not surprisingly, mailers have a very poor security record