## CSci 5271 Introduction to Computer Security Day 8: Defensive programming and design, part 2

Stephen McCamant University of Minnesota, Computer Science & Engineering

## Outline

### Secure use of the OS

Announcements intermission

Bernstein's perspective

Techniques for privilege separation

## Avoid special privileges

 Require users to have appropriate permissions

 Rather than putting trust in programs

 Anti-pattern 1: setuid/setgid program

- Anti-pattern 2: privileged daemon
- But, sometimes unavoidable (e.g., email)

## One slide on setuid/setgid

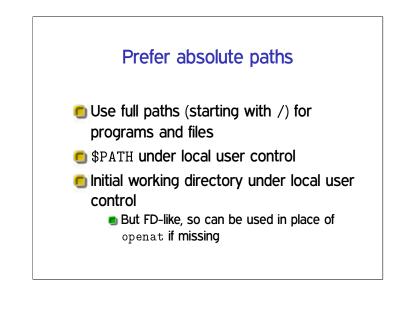
- Unix users and process have a user id number (UID) as well as one or more group IDs
- Normally, process has the IDs of the use who starts it
- A setuid program instead takes the UID of the program binary

# Don't use shells or Tcl

- … in security-sensitive applications
- String interpretation and re-parsing are very hard to do safely
- Eternal Unix code bug: path names with spaces

## Prefer file descriptors

- Maintain references to files by keeping them open and using file descriptors, rather than by name
- References same contents despite file system changes
- Use openat, etc., variants to use FD instead of directory paths



# Prefer fully trusted paths

- Each directory component in a path must be write protected
- Read-only file in read-only directory can be changed if a parent directory is modified

## Don't separate check from use

Avoid pattern of e.g., access then open
 Instead, just handle failure of open
 You have to do this anyway
 Multiple references allow races
 And access also has a history of bugs

## Be careful with temporary files

 Create files exclusively with tight permissions and never reopen them
 See detailed recommendations in Wheeler
 Not quite good enough: reopen and check matching device and inode
 Fails with sufficiently patient attack

# Give up privileges

Using appropriate combinations of set\*id functions

 Alas, details differ between Unix variants
 Best: give up permanently
 Second best: give up temporarily
 Detailed recommendations: Setuid Demystified (USENIX'02)

## Whitelist environment variables

- Can change the behavior of called program in unexpected ways
- Decide which ones are necessary As few as possible
- 🖲 Save these, remove any others

## Outline

Secure use of the OS

Announcements intermission

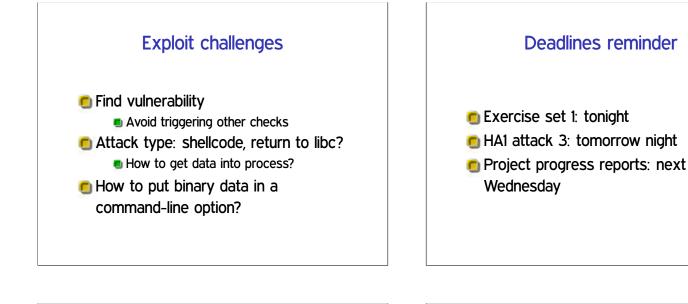
Bernstein's perspective

Techniques for privilege separation

## HA1 instructions

You may use any written source you can find to help with this assignment, on paper or the Internet, but you must explicitly reference any sources other than the lecture notes, assigned readings, and course staff.

Example: shellcode





Secure use of the OS

Announcements intermission

Bernstein's perspective

Techniques for privilege separation

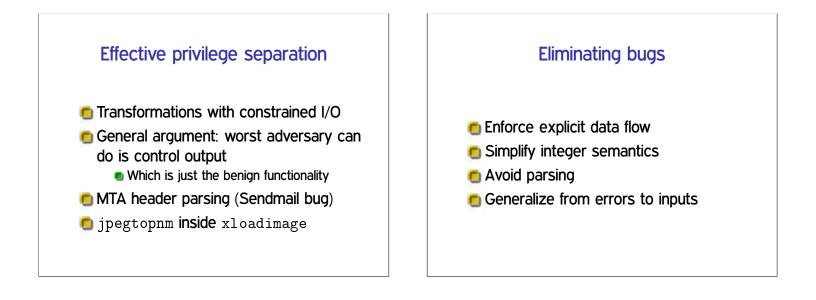


# Distinctive qmail features

- Single, security-oriented developer
- Architecture with separate programs and UIDs
- Replacements for standard libraries
- Deliveries into directories rather than large files

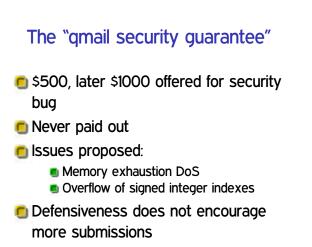
## Ineffective privilege separation

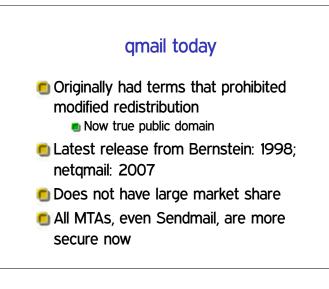
- Example: prevent Netscape DNS helper from accessing local file system
- Before: bug in DNS code
  - $\rightarrow\,$  read user's private files
- After: bug in DNS code
  - $\rightarrow\,$  inject bogus DNS results
  - $\rightarrow$  man-in-the-middle attack
  - ightarrow read user's private web data





- Identify common functions
- Automatically handle errors
- Reuse network tools
- Reuse access controls
- Reuse the filesystem





## Outline

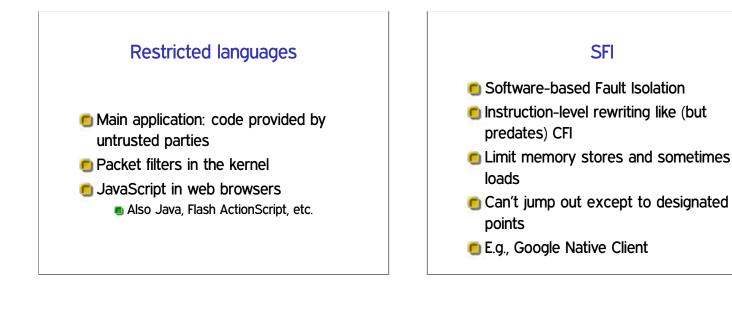
Secure use of the OS

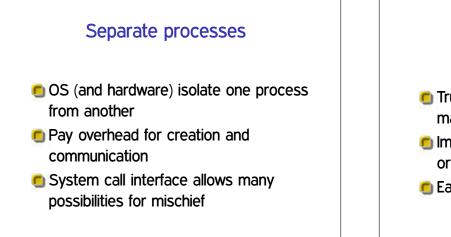
Announcements intermission

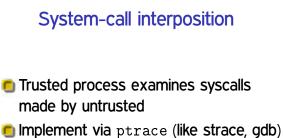
Bernstein's perspective

Techniques for privilege separation

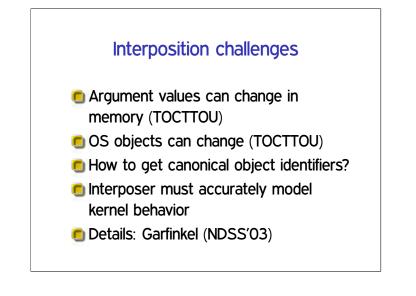
SFI





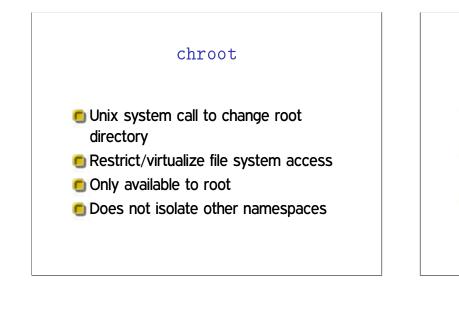


- or via kernel change
- Easy policy: deny



## Separate users

- Reuse OS facilities for access control
- 🖲 Unit of trust: program or application
- 🖲 Older example: qmail
- 🖲 Newer example: Android
- Limitation: lots of things available to any user



# **OS-enabled containers**

- One kernel, but virtualizes all namespaces
- FreeBSD jails, Linux LXC, Solaris zones, etc.
- Quite robust, but the full, fixed, kernel is in the TCB

