CSci 5271 Introduction to Computer Security Day 12: OS security: higher assurance

Stephen McCamant University of Minnesota, Computer Science & Engineering

Outline

Micro-architectural side channels

Announcements intermission

OS trust and assurance

More Unix permissions

Outline

Micro-architectural side channels

Announcements intermission

OS trust and assurance

More Unix permissions

Exercise Set 2 submissions

- We are using different Gradescope features for Exercise Set 2
- Create and submit a PDF document with your answers
 - Still prefer typed written answers, for readability
 LaTeX and Google Docs templates available

Gradescope submissions now available, include your group

Exercise Set 1 regrading

 This Wednesday will be the last day to request re-grades of Exercise Set 1 (via Gradescope)
 In particular, not entertained at the end of the semester

Outline

Micro-architectural side channels

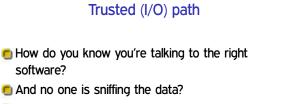
Announcements intermission

OS trust and assurance

More Unix permissions

Trusted and trustworthy

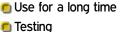
- Part of your system is trusted if its failure can break your security
- Thus, OS is almost always trusted
- Real question: is it trustworthy?
- Distinction not universally observed: trusted boot, Trusted Solaris, etc.



- 🖲 Example: Trojan login screen
 - Or worse: unlock screensaver with root password
 - Origin of "Press Ctrl-Alt-Del to log in"



How to gain assurance



- 🗖 Code / design review
- Third-party certification
- Formal methods / proof

Evaluation / certification

- Testing and review performed by an independent party
- 🖲 Goal: separate incentives, separate accountability
- Compare with financial auditing
- Watch out for: form over substance, misplaced incentives

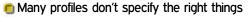
Orange book OS evaluation

- Trusted Computer System Evaluation Criteria
- D. Minimal protection
- C. Discretionary protection
- C2 adds, e.g., secure audit over C1
 B. Mandatory protection
 - B1<B2<B3: stricter classic MLS</p>
- A. Verified protection

Common Criteria

- International standard and agreement for IT security certification
- Certification against a protection profile, and evaluation assurance level EAL 1-7
- Evaluation performed by non-government labs
- Up to EAL 4 automatically cross-recognized

Common Criteria, Anderson's view



- OSes evaluated only in unrealistic environments E.g., unpatched Windows XP with no network attacks
- Corruption, Manipulation, and Inertia"
 - Pernicious innovation: evaluation paid for by vendor
 Labs beholden to national security apparatus

Formal methods and proof

- Can math come to the rescue?
- Checking design vs. implementation
- Automation possible only with other tradeoffs E.g., bounded size model
- Starting to become possible: machine-checked proof

Proof and complexity

- Formal proof is only feasible for programs that are small and elegant
- If you honestly care about assurance, you want your TCB small and elegant anyway
- Should provability further guide design?

Some hopeful proof results

seL4 microkernel (SOSP'09 and ongoing)

 7.5 kL C, 200 kL proof, 160 bugs fixed, 25 person years
 CompCert C-subset compiler (PLDI'06 and ongoing)
 RockSalt SFI verifier (PLDI'12)

Outline

Micro-architectural side channels

Announcements intermission

OS trust and assurance

More Unix permissions

"POSIX" ACLs

Based on a withdrawn standardization

- More flexible permissions, still fairly Unix-like
- Multiple user and group entries
 - Decision still based on one entry
- Default ACLs: generalize group inheritance
- Command line: getfacl, setfacl

ACL legacy interactions

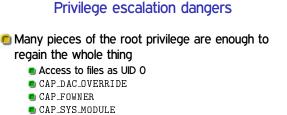
- Hard problem: don't break security of legacy code Suggests: "fail closed"
- Contrary pressure: don't want to break functionality
 Suggests: "fail open"
- POSIX ACL design: old group permission bits are a mask on all novel permissions

"POSIX" "capabilities"

- Divide root privilege into smaller (~35) pieces
- 🖲 Note: not real capabilities
- First runtime only, then added to FS similar to setuid
- 🖲 Motivating example: ping
- 🖲 Also allows permanent disabling

Legacy interaction dangers

Former bug: take away capability to drop privileges
 Use of temporary files by no-longer setuid programs
 For more details: "Exploiting capabilities", Emeric Nasi



- CAP_MKNOD
- CAP_PTRACE
- CAP_SYS_ADMIN (mount)