CSci 5271 Introduction to Computer Security Day 11: OS security: higher assurance

Stephen McCamant University of Minnesota, Computer Science & Engineering

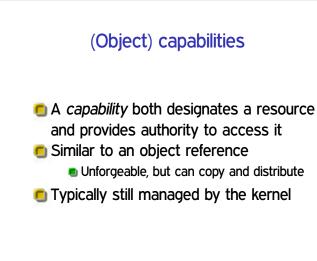
Outline

Capability-based access control

OS trust and assurance

Assignment debrief and announcements

More Unix access control



Capability slogans (Miller et al.)

- No designation with authority
- Dynamic subject creation
- 🖲 Subject-aggregated authority mgmt.
- No ambient authority
- Composability of authorities
- Access-controlled delegation
- 🖲 Dynamic resource creation



- Use indirection: give real capability via a pair of middlemen
- Retain capability to tell R to drop capability to B
- Depends on composability

Confinement with capabilities

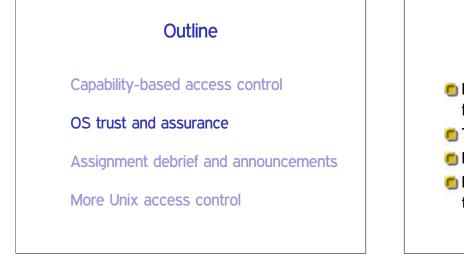
- A cannot pass a capability to B if it cannot communicate with A at all
- Disconnected parts of the capability graph cannot be reconnected
- Depends on controlled delegation and data/capability distinction

OKL4 and seL4

- Commercial and research microkernels
- Recent versions of OKL4 use capability design from seL4
- Used as a hypervisor, e.g. underneath paravirtualized Linux
- Shipped on over 1 billion cell phones

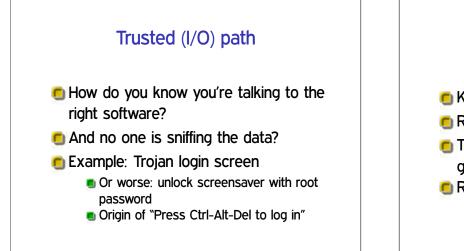
Joe-E and Caja

- Dialects of Java and JavaScript (resp.) using capabilities for confined execution
- E.g., of JavaScript in an advertisement
- Note reliance on Java and JavaScript type safety

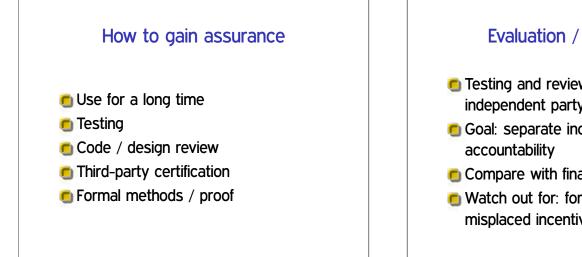




- Part of your system is trusted if its failure can break your security
- Thus, OS is almost always trusted
- Real question: is it trustworthy?
- Distinction not universally observed: trusted boot, Trusted Solaris, etc.







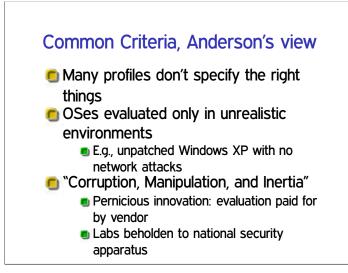
Evaluation / certification

- Testing and review performed by an independent party
- Goal: separate incentives, separate
- 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 A. Verified protection

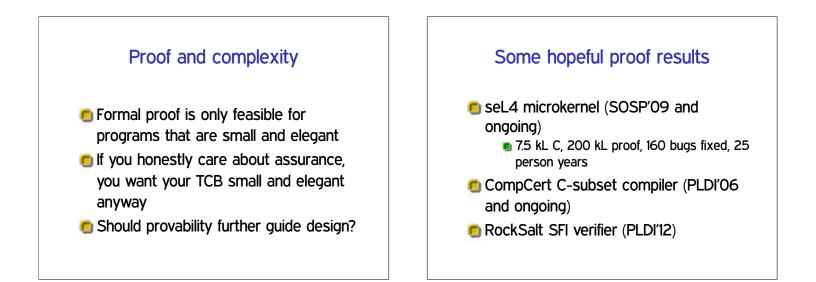
Common Criteria

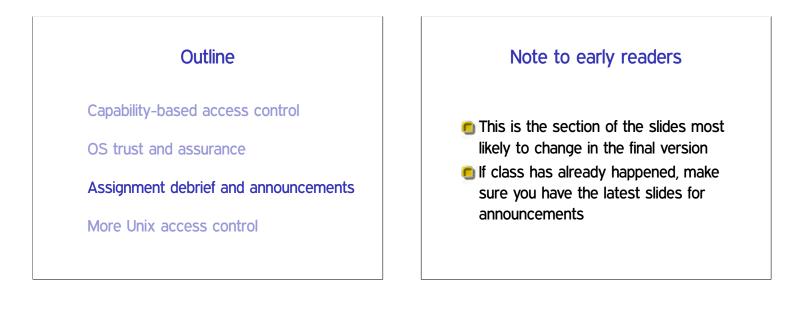
- International standard and agreement for IT security certification
- Certification against a protection profile, and evaluation assurance level EAL 1-7
- Evaluation performed by non-government labs
- 🖲 Up to EAL 4 automatically cross-recognized

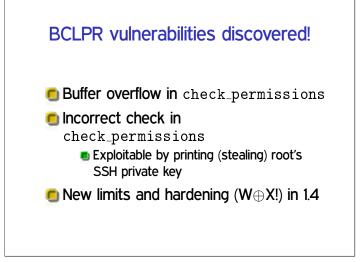


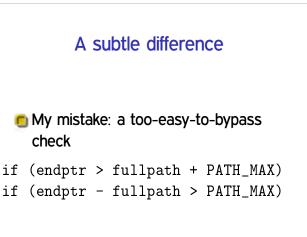
Formal methods and proof

- 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









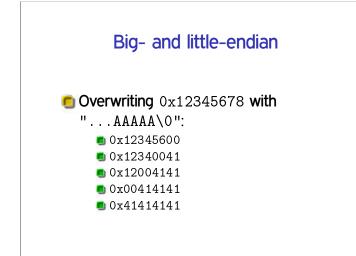
HA1 week 5

- Extra credit, for those who found weeks 1-4 too easy
- More subtle bugs and tricky exploit techniques
- Are there any bugs I didn't introduce on purpose?

Reversing the stack

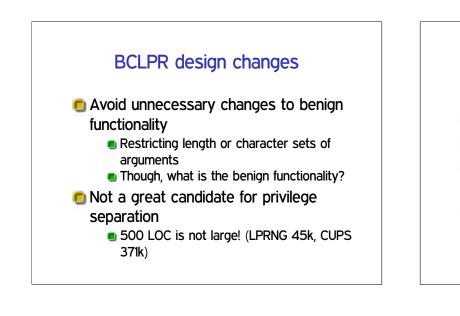
```
void func(char *str) {
   char buf[128];
   strcpy(buf, str);
   do_something();
   return;
}
```







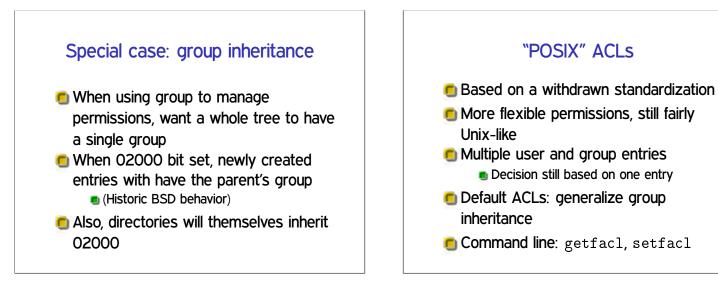
```
char *zip(char *a, char *b) {
   char *result;
   int len, i;
   len = strlen(a);
   result = malloc(2*len);
   for(i = 0; i <= len; i++) {
      result[2*i] = a[i];
      result[2*i+1] = b[i];
   }
   return result;
}</pre>
```

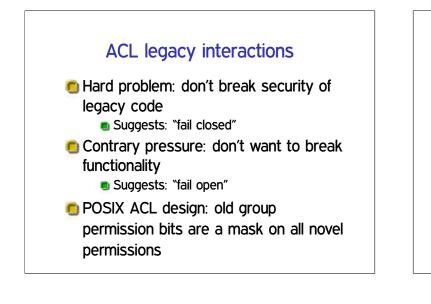


Midterm exam next Tuesday

- 🖲 Usual class time and location
- Covers up through today's lecture
- Mix of short-answer and exercise-like questions
- Open books/notes/printouts, no computers or other electronics

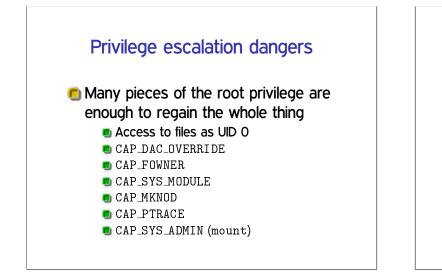






"POSIX" "capabilities"

- 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

Application: electronic voting

Next time