CSci 5271
Introduction to Computer Security
Capabilities, side channels, OS assurance

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Preview question

What's "common" about the Common Criteria?

A. Every kind of product is evaluated against the same "protection profile."
B. Anyone can perform the certification, without special government approval.
C. The certification applies to devices used in everyday civilian life, rather than in government or the military.
D. A single certification is recognized by the governments of many countries.
E. A single certification can be used for products from different vendors.

Outline

Capability-based access control (cont'd)

Side and covert channel basics

Announcements intermission

Transient execution covert channels

OS trust and assurance

(Object) capabilities

- A *capability* both designates a resource and provides authority to access it
- Similar to an object reference
  - Unforgeable, but can copy and distribute
- Typically still managed by the kernel

Capability slogans (Miller et al.)

- No designation without authority
- Dynamic subject creation
- Subject-aggregated authority management
- No ambient authority
- Composability of authorities
- Access-controlled delegation
- Dynamic resource creation

Partial example: Unix FDs

- Authority to access a specific file
- Managed by kernel on behalf of process
- Can be passed between processes
  - Though rare other than parent to child
- Unix not designed to use pervasively

Distinguish: password capabilities

- Bit pattern itself is the capability
  - No centralized management
- Modern example: authorization using cryptographic certificates

Revocation with capabilities

- Use indirection: give real capability via a pair of middlemen
- $A \to B$ via $A \to F \to R \to B$
- 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

Outline

Capability-based access control (cont'd)

Side and covert channel basics

Announcements intermission

Transient execution covert channels

OS trust and assurance

More confidentiality problems

- Careful access control prevents secret data from "leaking" though normal OS-mediated communication channels
- Residual problem: channels not designed for communication
- A major theme of ongoing computer security research

Side channel vs. covert channel

- Side channel: information leaks from an unsuspecting victim
- Covert channel: information intentionally leaked by a adversarial sender
  - Violating an isolation property
  - Sender and receiver work together
- Distinction sometimes unclear or not observed

Kinds of channels

- Software channels: undesired feature of program behaviors
- Physical channels: channels mediated by the real world
- Hardware channels: undesired feature of hardware behaviors

Classic software covert channels

- Storage channel: writable shared state
  - E.g., screen brightness on mobile phone
- Timing channel: speed or ordering of events
  - E.g., deliberately consume CPU time

Remote timing and traffic analysis

- Timing of events can also leak over the network
  - Classic example: time taken to process encrypted data
- Encrypted network traffic still reveals information via pattern and timing of packets
  - Classic example: keystrokes over SSH
  - Modern: "website fingerprinting" against HTTPS and Tor

Examples of physical side channels

- EM emissions and diffuse reflections from CRTs
- Power usage of computers and smart cards
- Smartphone accelerometer picks up speaker vibrations

Common hardware channel: cache timing

- Memory cache shared by processes and sometimes cores
- Cache state depends on pattern of previous accesses
- Cache hit or miss affects code execution speed

Outline

Capability-based access control (cont'd)

Side and covert channel basics

Announcements intermission

Transient execution covert channels

OS trust and assurance

Multiple BCMTA vulnerabilities found!

- Format string vulnerability in logging
- Race condition on file ownership check
- Instruction whitelist was too permissive

Midterm exam next Monday

- Usual class time and location
- Covers up through today's lecture material
- Mix of short-answer and exercise-like questions
- Open books/notes/printouts, no computers or other electronics
- Sample exams (2013-2019) posted, solutions Wednesday

Exercise set 2

- Due Wednesday evening
- Join pre-created groups in Canvas
- Remember to cite any outside sources you used
- May not be graded before midterm, so ask questions early

Reversing the stack

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

Payment app

```
void payment(char *name, double amount_jpy,
             char *purpose, int purpose_len) {
    double amount_usd = amount_jpy / 109.23;
    char memo[32];
    strcpy(memo, "Payment for: ");
    memcpy(memo + strlen(memo), purpose, purpose_len);
    write_check(name, amount_usd, memo);
}
```

Reverse range

```
void reverse_range(int *a, int from, int to) {
    unsigned int *p = &a[from];
    unsigned int *q = &a[to];
    while (!(p == q + 1 || p == q + 2)) {
        *p += *q;
        *q = *p - *q;
        *p = *p - *q;
        p++; q--;
    }
}
```

Outline

- Capability-based access control (cont'd)
- Side and covert channel basics
- Announcements intermission
- **Transient execution covert channels**
- OS trust and assurance

Outline

- Capability-based access control (cont'd)
- Side and covert channel basics
- Announcements intermission
- Transient execution covert channels
- **OS trust and assurance**

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.

Trusted (I/O) path

- How do you know you're talking to the right software?
- And no one is sniffing the data?
- Example: Trojan login screen
  - Or worse: unlock screensaver with root password
  - Origin of "Press Ctrl-Alt-Del to log in"

Minimizing trust

- Kernel $\rightarrow$ microkernel $\rightarrow$ nanokernel
- Reference monitor concept
- TCB size: measured relative to a policy goal
- Reference monitor $\subseteq$ TCB
  - But hard to build monitor for all goals

How to gain assurance

- Use for a long time
- Testing
- 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

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

- Many profiles don't specify the right things
- 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)