Lecture 4: Principles of Security

CS 181S

Fall 2020

Assurance

- **Security** = does what it should + nothing more
- This should be accompanied by an **assurance argument**, which is some compelling basis to believe the system is secure.

Axiomatic Trust

Analytic Trust

Synthetic Trust

- The set of system components that you have to trust in order for your security requirements to be satisfied is called the **Trusted Computing Base (TCB)**

Principle: Economy of Mechanism

Prefer mechanisms that are simpler and smaller

- Easier to construct, understand, analyze
- Hence less likely to have unknown vulnerabilities

- TCB should be small

Assumptions

- Attacker cannot replace/modify mechanism
- Attacker cannot circumvent mechanism

Enforcement Mechanisms

- An enforcement mechanism must either
  1) prevent the execution of those instructions or
  2) eliminate or mitigate the effects of those instructions

- Possible approaches to enforcement:
  1) Isolation
  2) Monitoring

Isolation

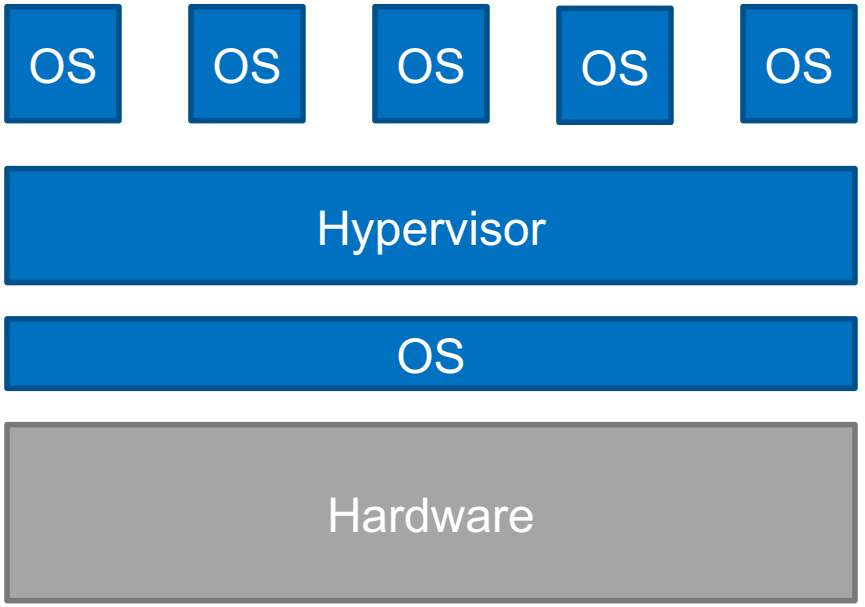
- Key idea: prevent or restrict the ability of one principal to influence execution by another

Isolation: Physical Isolation

- A Faraday cage is an enclosure made of conductive material or mesh
- The external electrical field causes electric charges to be distributed in a way that cancels out the field's effect on the interior
- This effectively blocks any sort of electromagnetic radiation
- US gov/military use rooms inside Faraday cages, called SCIFs, to hold classified meetings

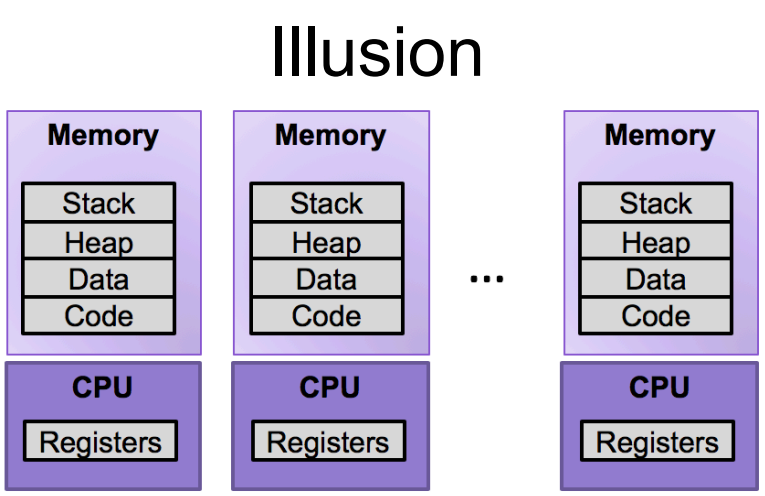
Isolation: Virtual Machines

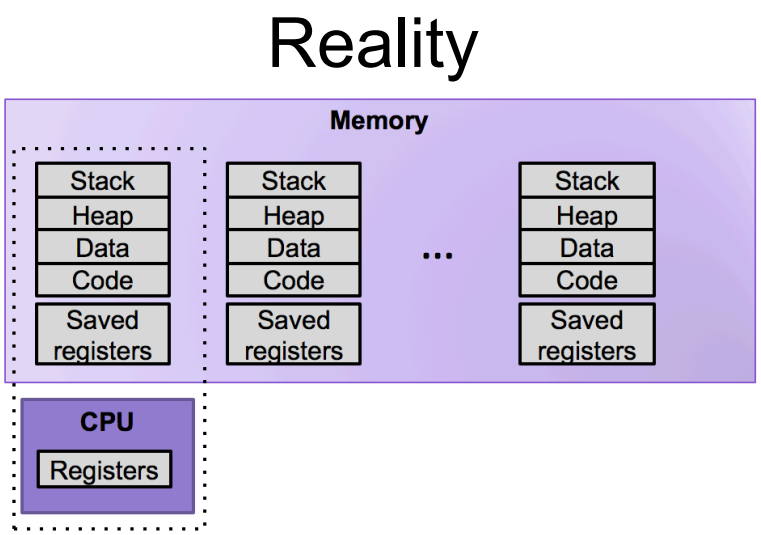
- A virtual machine behaves as if it were an isolated computer despite other execution on the underlying hardware
- A hypervisor implements virtual machines that have the same instruction set as the underlying hardware



Isolation: Processes

- The OS kernel multiplexes the actual processor and creates a set of processes
- Each process executes in its own isolated address space
- Kernel-supported instructions provide access to system services and shared resources





Isolation: Sandboxes

- A sandbox runs an application in a restricted environment
- Example: in Chrome, the entire HTML rendering and JavaScript execution is sandboxed (cannot access files or windows outside the current job, cannot read/write to clipboard)

Partial Isolation: Firewalls

- Idea: just enough isolation to block communications used for attacks
- A firewall interrupts the connection from some group of computers to some network
- It is configured to pass only certain messages (e.g., those to specific ports or from particular sources)

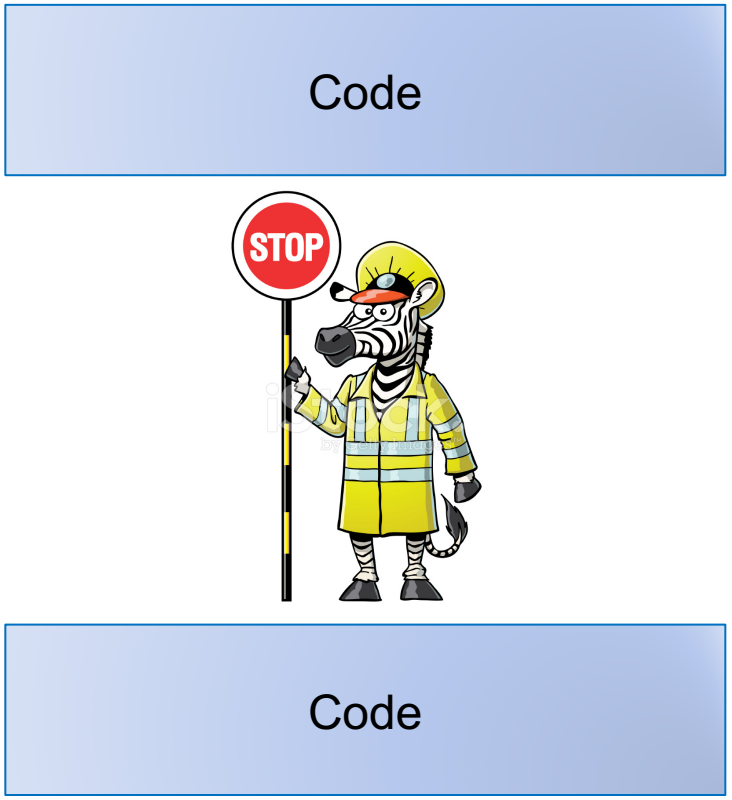
Partial Isolation: Code Signing

- Only code that is digitally signed by a trusted principal is allowed to execute
- Example: Microsoft Authenticode protects against web pages containing malicious executable content by only allowing downloaded content to be executed if it was produced by Microsoft or a Microsoft-approved software provider

Monitoring

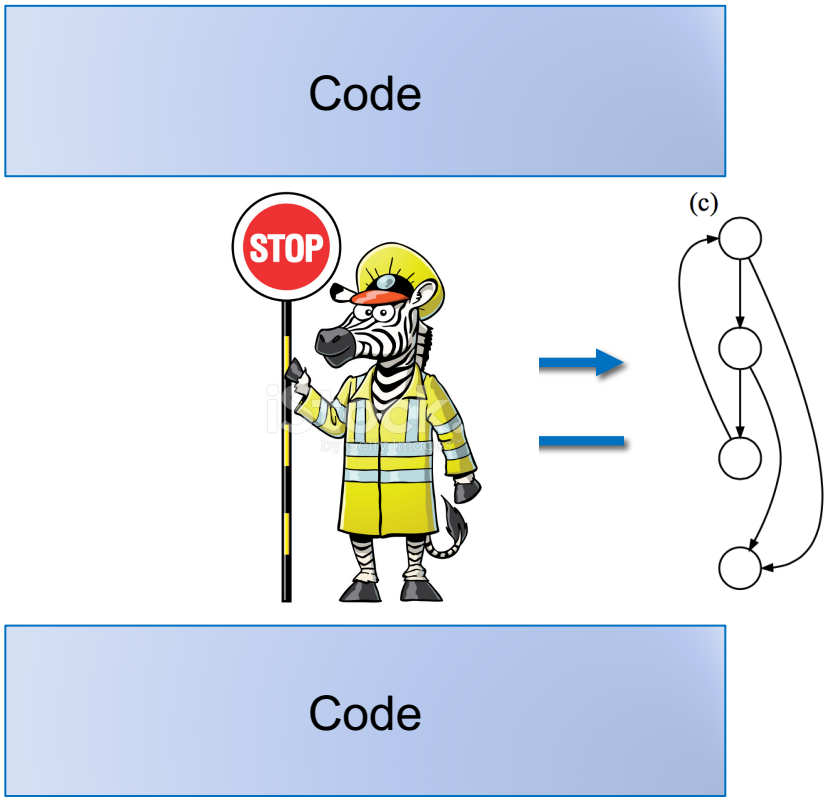
- Key idea: monitor a set of interfaces and halt malicious execution before any damage is done

- a **security policy** that describes acceptable sequences of operations
- a **reference monitor** that receives control whenever operations are requested
- a means by which the reference monitor can **block** further execution that does not comply with the policy



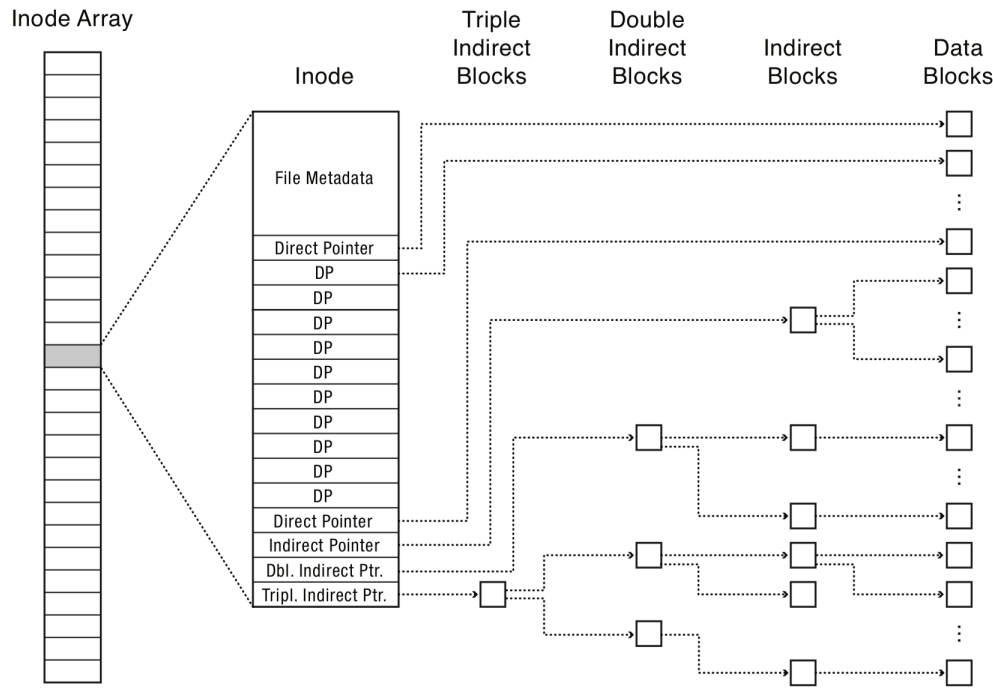
Monitoring: Control Flow Integrity

- Example: Control Flow Guard
- Approximate CFI implementation in Windows
- Jump is valid if it beginning of function
  - Granularity: 8 bytes
- Check implemented as bitmap



Monitoring: File Access Control

- inode for each file stores permission bits (r, w, x)
- Operating system enforces access control when file is is opened
- Error if principal is not authorized



Principle: Complete Mediation

Every operation requested by a principal must be intercepted and determined to be acceptable according to the security policy

- Component that does the interception and determination is the reference monitor
- Restricts caching of information, including previous decisions

Exercise 1: Complete Mediation

- Consider the security mechanisms deployed in your dorm or home. These systems are designed to prevent access by unauthorized people.

  - To what extent do those security features enforce Complete Mediation?

Principle: Least Privilege

Principals should be given the minimum privileges necessary to accomplish their task

- Limits the damage that can result from accident or malice
- Cf. "need to know"

Exercise 2: Least Privilege

- Consider the security mechanisms deployed in your dorm or home. These systems are designed to prevent access by unauthorized people.

  - To what extent do those security features enforce Least Privilege?

Principle: Separation of Privilege

- Different operations should require different privileges
- Disseminate privileges for an operation amongst multiple principals (Separation of Duty)

```
drwxr-xr-x   5 eleanor  staff     160 Mar 21 12:14 .
drwx------+ 54 eleanor  staff    1728 Mar 21 09:45 ..
-rw-r--r--@  1 eleanor  staff   98971 Mar 21 05:15 download.png
-rwxr-xr-x   1 root     wheel  103632 Mar 21 12:14 java
-r--------@  1 eleanor  staff    2085 Mar 21 12:07 rsa-demo.pem
```

Principle: Failsafe Defaults

Base decisions on the presence of privilege, not the absence of prohibition

- The default answer is "no"
- Say "yes" only when there is an explicit reason to do so
- Principals who discover they don't have access will complain
- Attackers who discover they do have access won't complain!

Principles of Prevention

[Saltzer and Schroeder, *The Protection of Information in Computer Systems*, 1975]

- Accountability
- Complete Mediation
- Least Privilege
- Failsafe Defaults
- Separation of Privilege
- Defense in Depth
- Economy of Mechanism
- Open Design

Principle: Defense in Depth

Prefer a set of complementary mechanisms over a single mechanism

Complementary:

- **Independent:** attack that compromises one mechanism is unlikely to compromise others
- **Overlapping:** attacks must compromise multiple mechanisms to succeed

Exercise 3: Defense in Depth

- Consider the security mechanisms deployed in your dorm or home. These systems are designed to prevent access by unauthorized people.

  - To what extent do those security features satisfy the overlapping requirement of Defense in Depth?
  - How could the security features be modified to add defense in depth if it does not already exist?

Principle: Open Design

Security shouldn't depend upon the secrecy of design or implementation


```
/*     efdtt.c     Author:  Charles M. Hannum <root@ihack.net>                */
#define m(i)(x[i]^s[i+84])<<
unsigned char x[5],y,s[2048];main(n){for(read(0,x,5);read(0,s,n=2048);write(1,s
,n))if(s[y=s[13]%8+20]/16%4==1){int i=m(1)17^256+m(0)8,k=m(2)0,j=m(4)17^m(3)9^k
*2-k%8^8,a=0,c=26;for(s[y]-=16;--c;j*=2)a=a*2^i&1,i=i/2^j&1<<24;for(j=127;++j<n
;c=c>y)c+=y=i^i/8^i>>4^i>>12,i=i>>8^y<<17,a^=a>>14,y=a^a*8^a<<6,a=a>>8^y<<9,k=s
[j],k="7Wo~'G_\216"[k&7]+2^"cr3sfw6v;*k+>/n."[k>>4]*2^k*257/8,s[j]=k^(k&k*2&34)
*6^c+~y;}}
```

Principle: Open Design

Security shouldn't depend upon the secrecy of design or implementation

Arguments for open design:

- Secrets eventually come out: reverse engineering is possible, employees move around
- Making details public increases chance of identifying and repairing vulnerabilities

Principle: Open Design

Security shouldn't depend upon the secrecy of design or implementation

Arguments against open design:

- Secrecy supports Defense in Depth by making it harder to find vulnerabilities
- Lack of hard evidence that Linus' Law really holds ("given enough all eyeballs, all bugs are shallow")
- After identification, some vulnerabilities cannot quickly or easily be repaired

Exercise 4: Open Design

- Briefly (1-2 sentences) argue why you believe a system should or should not follow the principle of open design.

Countermeasures

A defense that protects against attacks by neutralizing either the threat or vulnerability involved

Strategy:

- Prevent: block attack or close vulnerability — Prevention
- Deter: make attack harder
- Deflect: make other targets more attractive — Risk Management
- Mitigate: make harm less severe
- Detect: as it happens or after the fact — Deterrence through Accountability
- Recover: undo harm

Exercise 5: Feedback

1. Rate how well you think this recorded lecture worked
    1. Better than an in-person class
    2. About as well as an in-person class
    3. Less well than an in-person class, but you still learned something
    4. Total waste of time, you didn't learn anything

2. How much time did you spend on this video lecture (including time spent on exercises)?

3. Do you have any comments or feedback?

Priniciples of Security