

OS Security Basics

CS642:

Computer Security



Professor Ristenpart

<http://www.cs.wisc.edu/~rist/>

rist at cs dot wisc dot edu

Arizona Botnet Controller Draws 30-Month Federal Sentence

Posted by **timothy** on Sunday September 09, @05:35PM
from the such-a-sweet-boy dept.



dgharmon writes with word from the BBC that

"A U.S. hacker who sold access to thousands of hijacked home computers has been jailed for 30 months. Joshua Schichtel of Phoenix, Arizona, was sentenced for renting out more than 72,000 PCs that he had taken over using computer viruses."

Time is cheap: Schichtel admitted to giving access to those 72,000 computers for \$1500.

We start with some basics about operating system security:



Multics

Multi-level security

Security policies

Access controls

UNIX permissions

Take yourself back to the 1960's



<http://fyeahhippies.tumblr.com/post/135907376>

Take yourself back to the 1960's

Time-share multiuser
computers coming into
use

GE-645

36 bit address space

Up to 4 processors

Magnetic tape drives

Supported virtual memory in hardware



Courtesy of
<http://aficionadous.blogspot.com/>

Multiplexed Information and Computing Service (Multics)

Project to develop operating system for time-shared systems

- Designed from 1964-1967.
- MIT project MAC, Bell Labs, and GE
- ~100 installations at greatest extent
- Last one shut down in 2000 (Canadian department of defense)

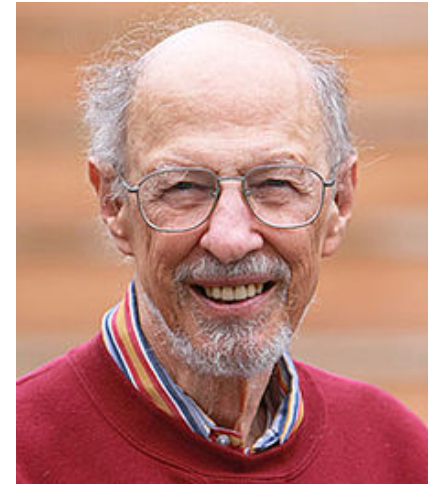
“A small but useful hardware complement would be 2 CPU units, 128K of core, 4 million words of high speed drum, 16 million words of disc, 8 tapes, 2 card readers, 2 line printers, 1 card punch and 30 consoles.”

[Vyssotsky, Corbato, Graham 1965]

Multics: ancestor to many OS's

Lots of innovations in design

- Use of segmentation and virtual memory with hardware support
- SMP (shared memory multiprocessor)
- Written in PL/1 (high level language)

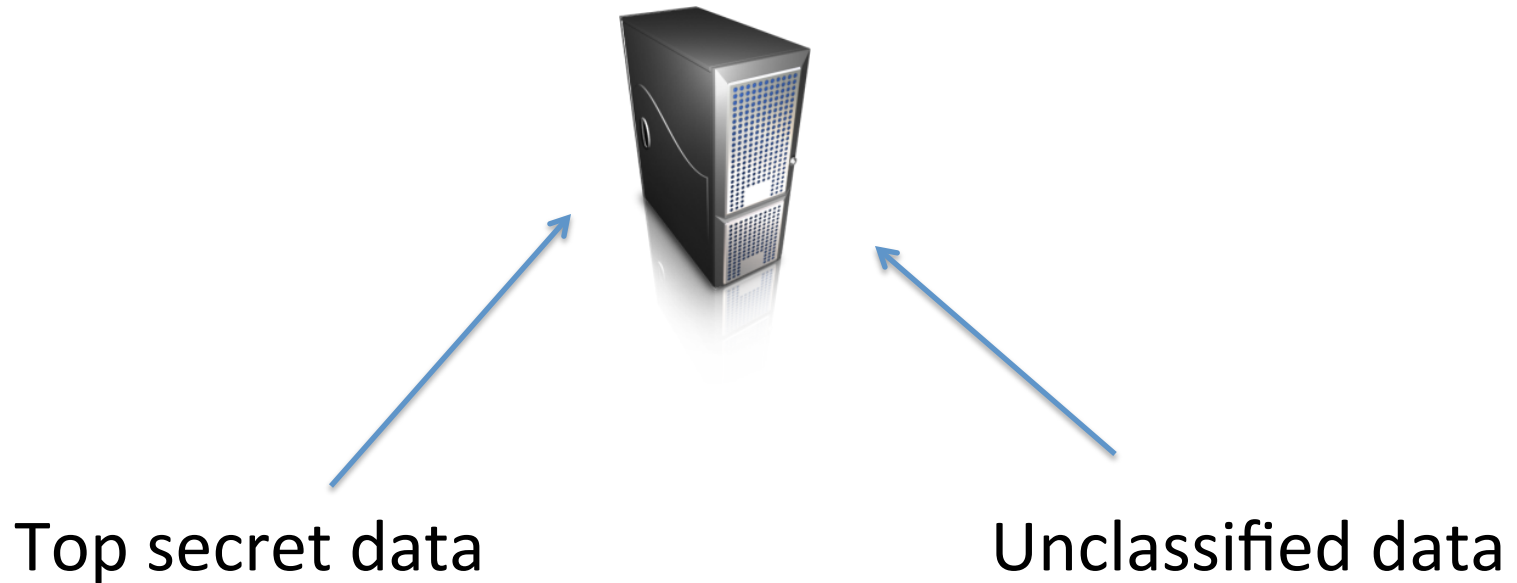


F. Corbato, MIT

Significant attention paid to security

Multi-level security

- Military and other government entities want to use time-sharing too



Classification levels

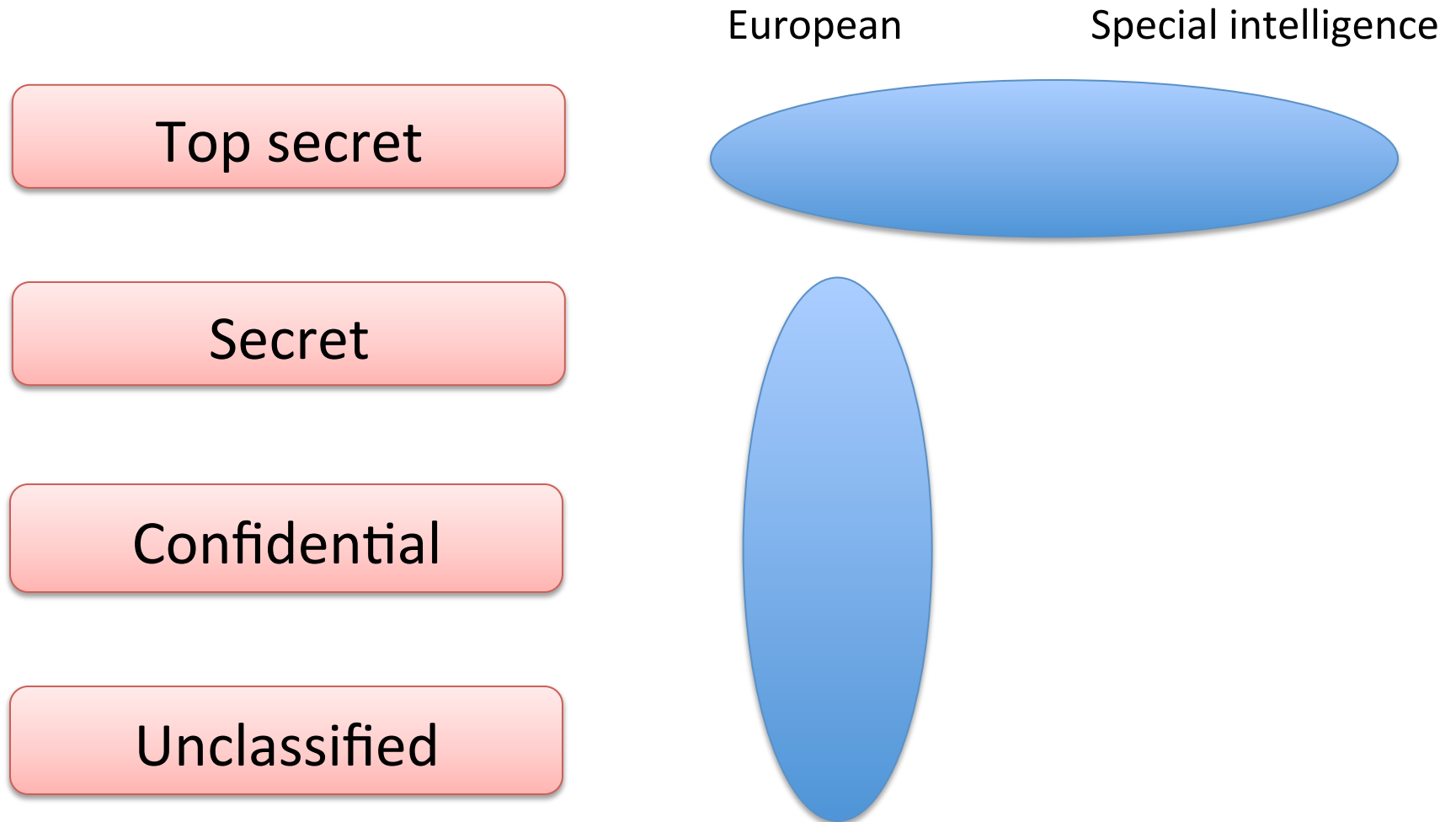
Top secret

Secret

Confidential

Unclassified

Classification levels and compartmentalization



Classification levels and compartmentalization

- Security level (L,C)
 - L is classification level (Top secret, secret, ...)
 - C is compartment (Europe, Special intelligence...)

Dominance relationship:

$$(L1, C1) \leq (L2, C2)$$

$L1 < L2$ ($L1$ “less secret” than $L2$)

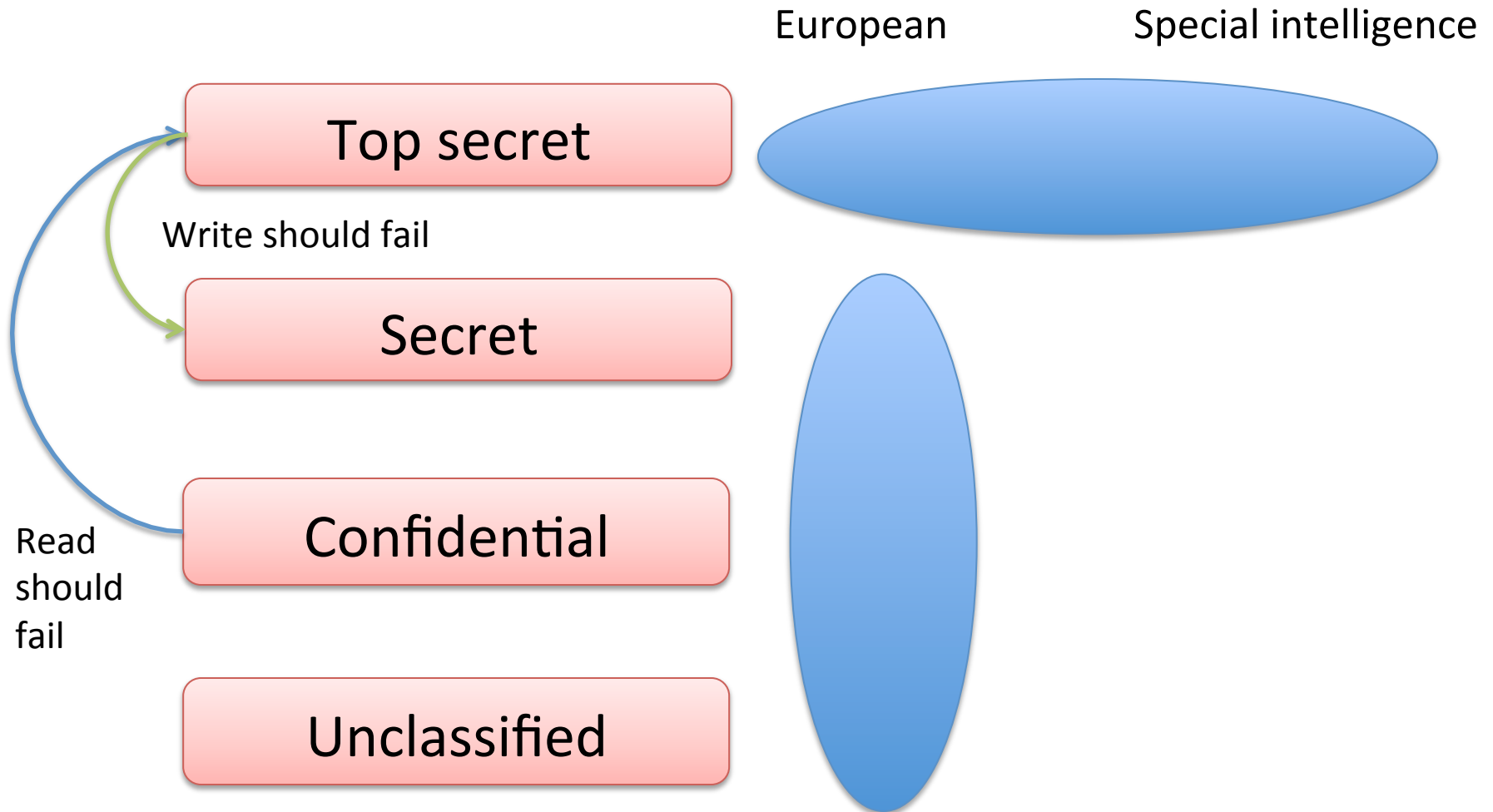
$C1$ subset of $C2$

Example:

$$(\text{Secret}, \{\text{European}\}) \leq (\text{Top Secret}, \{\text{European}, \text{Special Intel}\})$$

Bell-LaPadula Confidentiality Model

“no reads up”, “no writes down”



Bell-LaPadula Confidentiality Model

“no reads up”, “no writes down”

Simple security condition

User with $(L1, C1)$ can read file with $(L2, C2)$ if?

~~$(L1, C1) \leq (L2, C2)$~~ or $(L1, C1) \geq (L2, C2)$

*-property

User with $(L1, C1)$ can write file with $(L2, C2)$ if?

$(L1, C1) \leq (L2, C2)$ or ~~$(L1, C1) \geq (L2, C2)$~~



Say we have just Bell-Lapadula in effect... what could go wrong?

Biba integrity model

“no read down”, “no writes up”

European

Special intelligence

Top secret

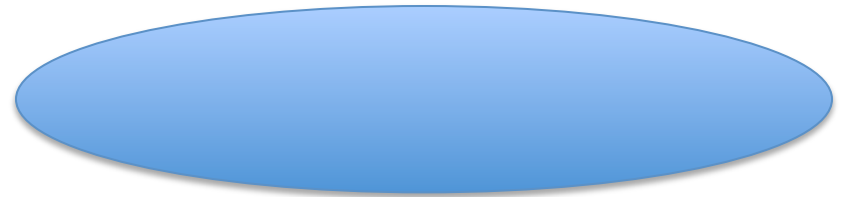
Read should fail

Secret

Confidential

Write
should
fail

Unclassified



Biba integrity model

“no read down”, “no writes up”

Simple integrity condition

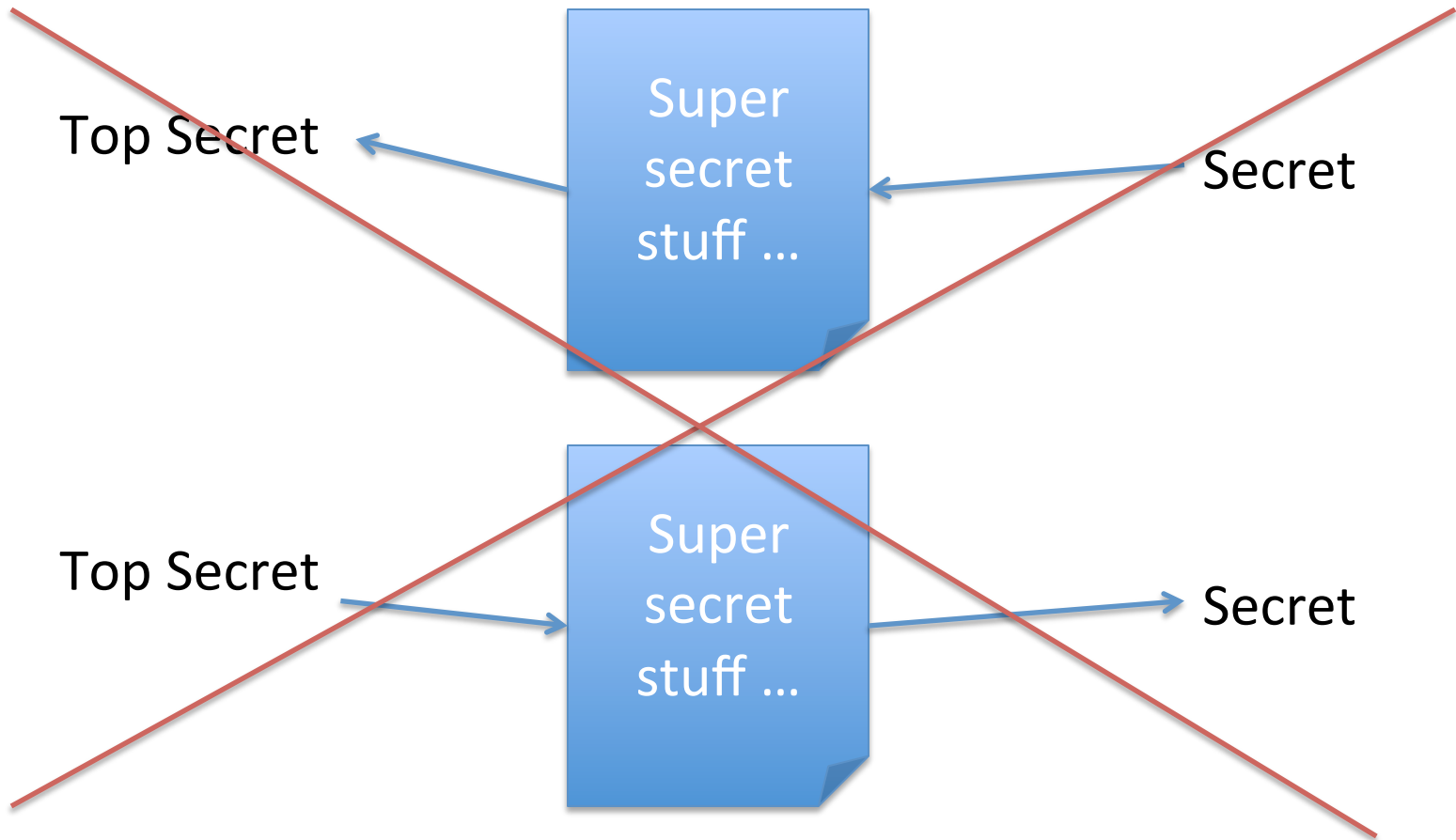
User with $(L1, C1)$ can read file with $(L2, C2)$ if?

$(L1, C1) \leq (L2, C2)$ or ~~$(L1, C1) \geq (L2, C2)$~~

*-property

User with $(L1, C1)$ can write file with $(L2, C2)$ if

~~$(L1, C1) \geq (L2, C2)$~~ or $(L1, C1) \geq (L2, C2)$



If we combine them... one can only communicate in same classification

Other policy models

- Capability model
- Decentralized information flow control
- Take-grant protection model
- Chinese wall
- Clarke-Wilson integrity model

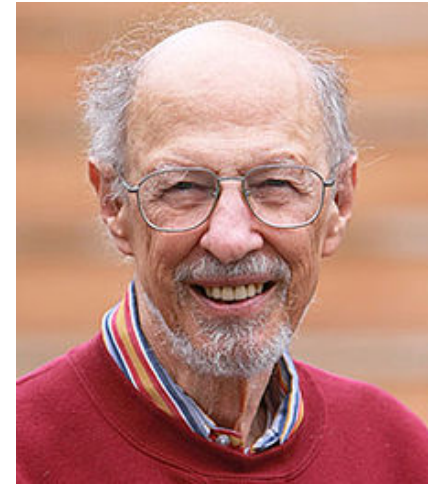
A good reference is:

Bishop, Computer Security: Art and Science

Multics: ancestor to many OS's

Lots of innovations in design

- Use of segmentation and virtual memory with hardware support
- SMP (shared memory multiprocessor)
- Written in PL/1 (high level language)



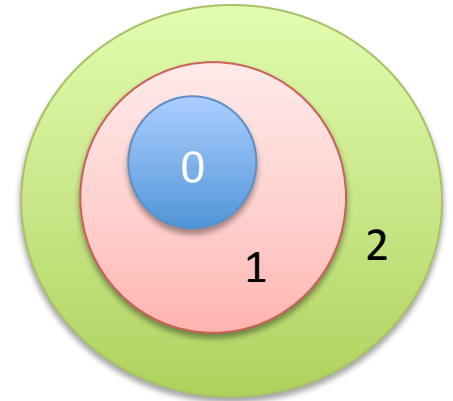
F. Corbato, MIT

Significant attention paid to security

Multics: security mechanisms

Protection rings 0-7
in which processes execute

- Lower number = higher privilege
- Ring 0 is “hardcore” supervisor
- Inherit privileges over higher levels

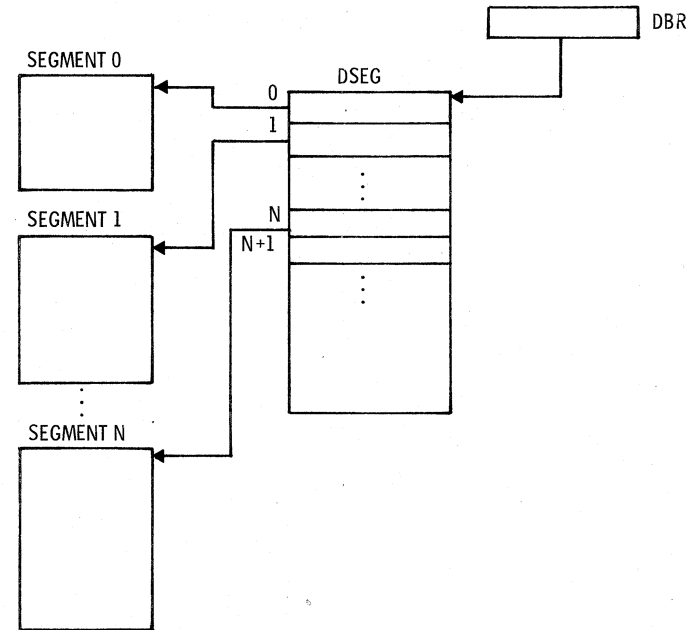


Protection rings included in all typical CPUs today and used by most operating systems

Multics: security mechanisms

Segments

- Virtual memory
- Program and data items stored in a segment
- Descriptor control field (read only, write only, execute only, ...)
- Segments access controlled



Multics: security mechanisms

Enciphered passwords

pw = 12345

- “I was no cryptanalyst ... Joe [Weizenbaum] had suggested I store the square of the password, but I knew people could take square roots, so I squared and ANDed with a mask to discard some bits.” – T. Van Vleck
- Later ones used DES, but Multics predates DES

pw  h(pw)



Karger and Schell

multicians.org

Karger and Schell: security analysis of Multics

- Classic red teaming example

We have concluded that AFDSC cannot run an open multi-level secure system on Multics at this time. As we have seen above, a malicious user can penetrate the system at will with relatively minimal effort. However, Multics does provide AFDSC with a basis for a benign multi-level system in which all users are determined to be trustworthy to some degree. For example, with certain enhancements, Multics could serve AFDSC in a two-level security mode with both Secret and Top Secret cleared users simultaneously accessing the system. Such a system, of course, would depend on the administrative determination that since all users are cleared at least to Secret, there would be no malicious users attempting to penetrate the security controls.

Karger and Schell: security analysis of Multics

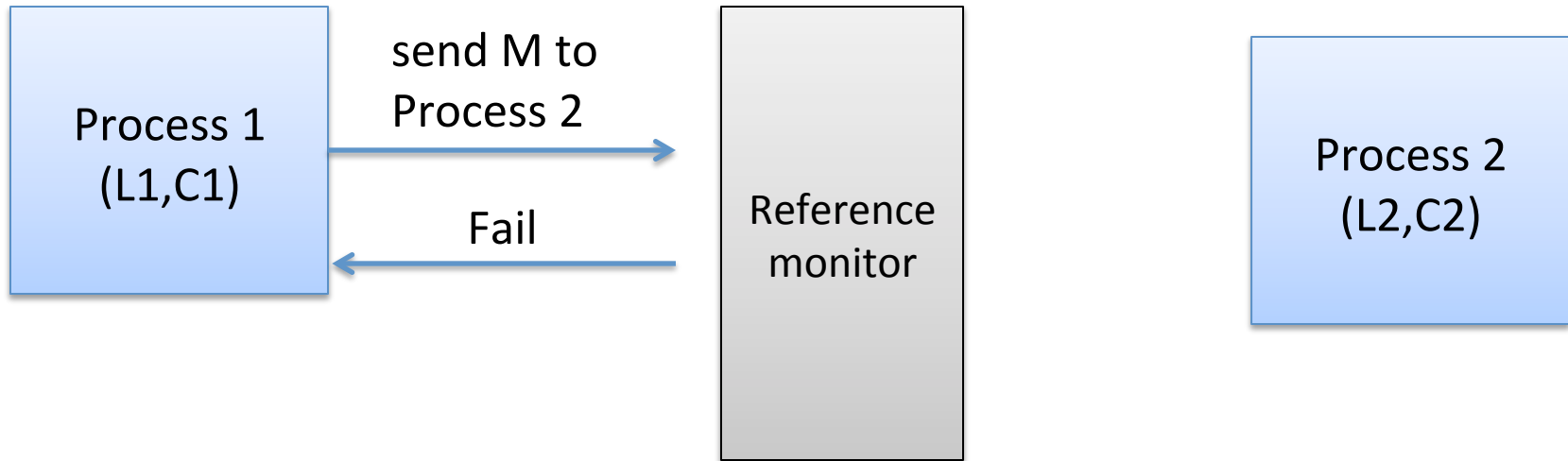
In the long term, it is felt that Multics can be developed into an open secure multi-level system by restructuring the operating system to include a security kernel. Such restructuring is essential since malicious users cannot be ruled out in an open system. The

Reference monitors / security kernels

- System component that monitors (hopefully all) accesses to data for security violations
- Reference monitors may be:
 - kernel
 - hypervisor
 - within applications (Apache)

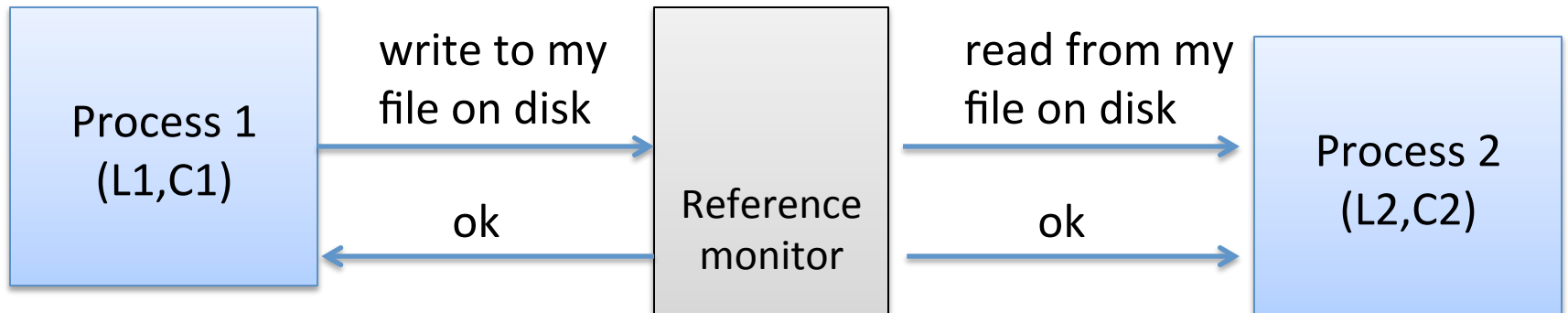
Circumventing access controls: covert channels

$$(L1, C1) \geq (L2, C2)$$



Circumventing access controls: covert channels

$$(L1, C1) \geq (L2, C2)$$



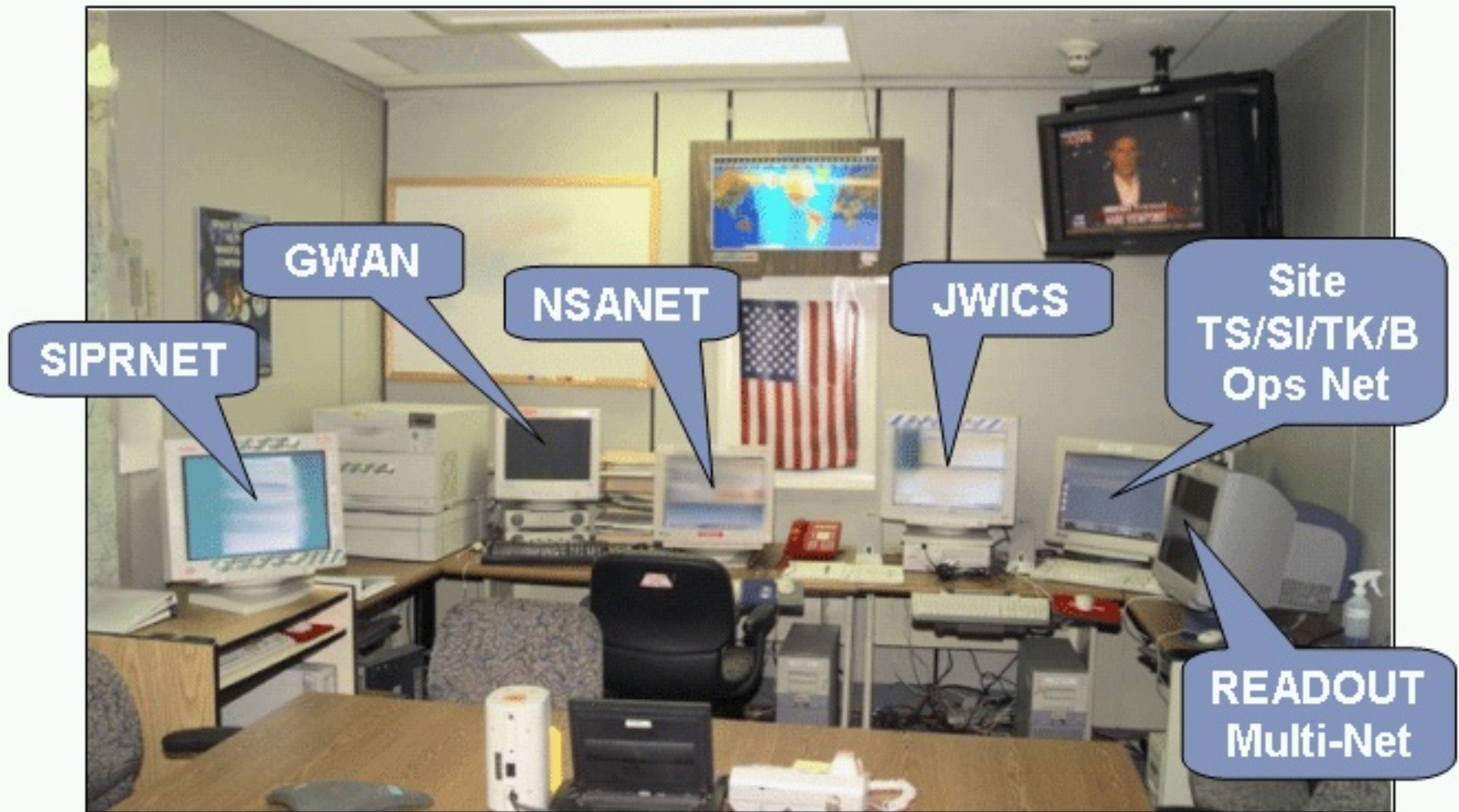
Process 1 sends a 1 bit to Process 2 by writing lots of bits to files it controls on hard disk

Process 1 sends a 0 bit by idling

Process 2 measures time to read from its files on disk

Longer read time = 1 bit sent
Shorter read time = 0 bit sent

Covert channels one reason shared MLS systems unsolved problem



Access controls

Access control matrix

Objects

	file 1	file 2	...	file n
user 1	read, write	read, write, own		read
user 2				
...				
user m	append	read, execute		read,write, own

Subjects

User i has permissions for file j as indicated in cell $[i,j]$

Due originally to Lampson in 1971

Two common implementation paradigms

	file 1	file 2	...	file n
user 1	read, write	read, write, own		read
user 2				
...				
user m	append	read, execute		read,wr ite,own

(1) Access control lists

Column stored with file

(2) Capabilities

Row stored for each user

Unforgeable tickets given to user

ACLs compared to Capabilities

ACLs requires
authenticating user

Processes must be given
permissions

Reference monitor must
protect permission setting

Token-based approach
avoids need for auth

Tokens can be passed
around

Reference monitor must
manage tokens

UNIX-style file system

```
rist@seclab-laptop1.local: ~/work/revindiff/full — less — 80x24
total 27648
drwxr-xr-x 51 rist staff 1734 Aug 23 13:11 .
drwxr-xr-x 46 rist staff 1564 Jul 5 12:37 ..
drwxr-xr-x 7 rist staff 238 Jun 22 18:29 .svn
-rw-r--r-- 1 rist staff 321 Jun 2 22:38 Makefile
-rwxr-xr-x 1 rist staff 258319 May 11 00:18 abbrev.bib
-rwxr-xr-x 1 rist staff 242609 May 11 00:18 abbrev_short.bib
-rw-r--r-- 1 rist staff 3049 Jun 20 14:22 abstract.tex
-rw-r--r-- 1 rist staff 6921 May 11 00:18 accents.sty
-rw-r--r-- 1 rist staff 534 Jun 20 16:30 acknowledgements.tex
-rw-r--r-- 1 rist staff 535 Jun 4 14:49 acknowledgements.tex.bak
-rw-r--r-- 1 rist staff 1813843 Jun 1 16:50 blah.zip
-rw-r--r-- 1 rist staff 2150 Jun 4 14:13 citesort.sty
-rwxr-xr-x 1 rist staff 30 May 11 00:18 conf.bib
-rw-r--r-- 1 rist staff 1321 May 11 00:18 cornercase.tex
-rw-r--r-- 1 rist staff 1385 May 11 00:18 crpproof.tex
-rwxr-xr-x 1 rist staff 6927118 May 11 00:18 crypto.bib
-rw-r--r-- 1 rist staff 59648 Jun 22 15:27 defs.tex
-rw-r--r-- 1 rist staff 1115 May 11 00:18 entropymeasures.tex
-rw-r--r-- 1 rist staff 10634 May 11 00:18 extattacks.tex
-rw-r--r-- 1 rist staff 815 May 11 00:18 extattcounterexample.tex
-rw-r--r-- 1 rist staff 8597 May 11 00:18 failedhashprop.tex
-rw-r--r-- 1 rist staff 11355 Jun 22 15:08 gamebased.tex
:
```


Who uses capabilities?

- Amoeba: distributed operating system (1990's)
- Eros (extremely reliable operating system)



- IBM System 38
- Intel iAPX 432

Capabilities are used in various ways inside modern systems all over

(From Wikipedia)

Delegation

Need to give a process, other user access

In ACL, process run by user inherits user's permissions

In Cap, process can pass around token

Revocation

Take away access from user or process

In ACL, remove user from list

In Cap, more difficult

Reference monitor must know where tokens are

Using pointer indirection

Roles (groups)

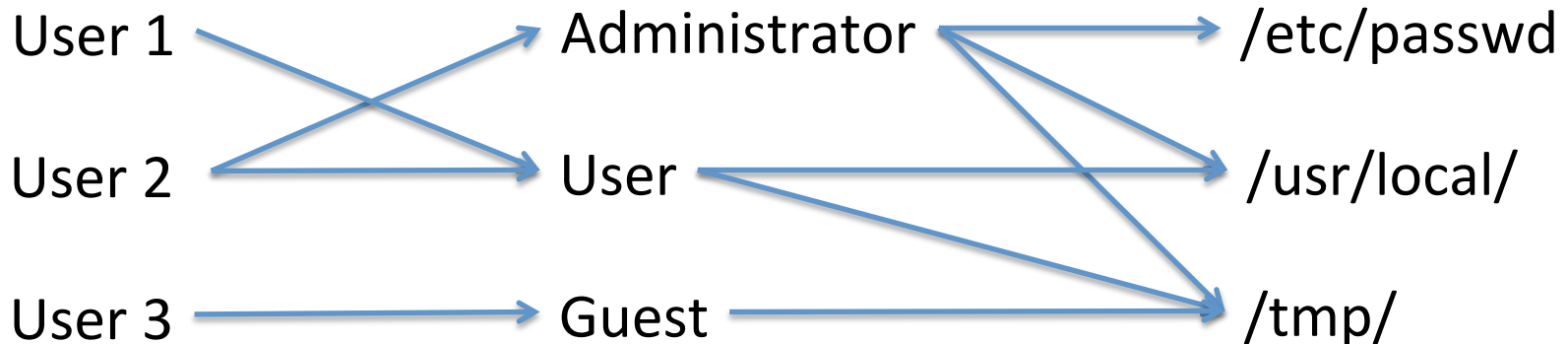
Group is a set of users

Administrator

User

Guest

Simplifies assignment of permissions at scale

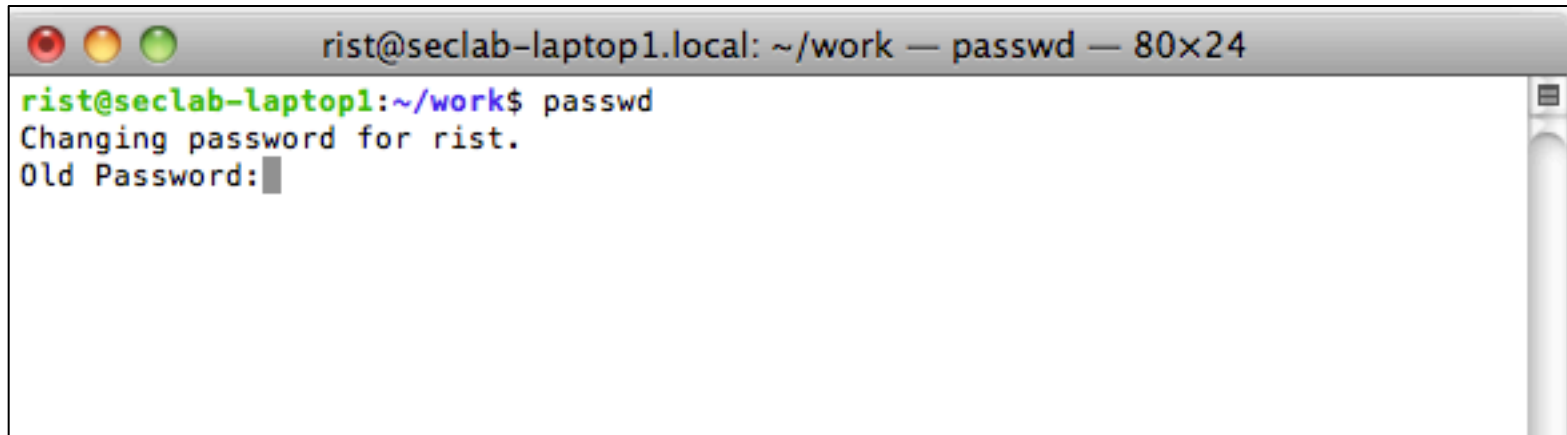


UNIX file permissions

- Owner, group
- Permissions set by owner / root
- Resolving permissions:
 - If user=owner, then owner privileges
 - If user in group, then group privileges
 - Otherwise, all privileges

UNIX Process permissions

- Process (normally) runs with permissions of user that invoked process

A terminal window with a title bar that reads "rist@seclab-laptop1.local: ~/work — passwd — 80x24". The terminal content shows the user "rist" at the prompt "rist@seclab-laptop1:~/work\$" typing the command "passwd". The output shows "Changing password for rist." followed by "Old Password:" and a cursor. The terminal window has standard macOS window controls (red, yellow, green buttons) and a scrollbar on the right side.

```
rist@seclab-laptop1.local: ~/work — passwd — 80x24
rist@seclab-laptop1:~/work$ passwd
Changing password for rist.
Old Password:█
```

/etc/shadow is owned by root

Users shouldn't be able to write to it generally



```
-r-xr-xr-x 1 root wheel 50512 Feb 10 2011 yes
-r-xr-xr-x 1 root wheel 50832 Feb 10 2011 ypcat
-r-xr-xr-x 1 root wheel 50864 Feb 10 2011 ypmatch
-r-xr-xr-x 1 root wheel 55344 Feb 10 2011 ypwhich
-rwxr-xr-x 2 root wheel 146976 Feb 10 2011 zcat
-rwxr-xr-x 1 root wheel 71 Feb 10 2011 zcmp
-rwxr-xr-x 1 root wheel 4422 Feb 10 2011 zdiff
-rwxr-xr-x 1 root wheel 66 Feb 10 2011 zegrep
-rwxr-xr-x 1 root wheel 66 Feb 10 2011 zfgrep
-rwxr-xr-x 1 root wheel 2017 Feb 10 2011 zforce
-rwxr-xr-x 1 root wheel 4894 Feb 10 2011 zgrep
-rwxr-xr-x 1 root wheel 359968 Feb 10 2011 zip
-rwxr-xr-x 1 root wheel 168432 Feb 10 2011 zipcloak
-rwxr-xr-x 1 root wheel 1188 Feb 10 2011 zipgrep
-rwxr-xr-x 2 root wheel 265392 Feb 10 2011 zipinfo
-rwxr-xr-x 1 root wheel 155440 Feb 10 2011 zipnote
-rwxr-xr-x 1 root wheel 159632 Feb 10 2011 zipsplit
-rwxr-xr-x 1 root wheel 1735 Feb 10 2011 zless
-rwxr-xr-x 1 root wheel 2441 Feb 10 2011 zmore
-rwxr-xr-x 1 root wheel 4954 Feb 10 2011 znew
-r-xr-xr-x 1 root wheel 63424 Apr 29 17:30 zprint
```

```
rist@seclab-laptop1:/usr/bin$ ls -al passwd
```

```
-r-sr-xr-x 1 root wheel 111968 Apr 29 17:30 passwd
```

```
rist@seclab-laptop1:/usr/bin$ █
```

Process permissions continued

UID 0 is root

Real user ID (RUID) --

same as UID of parent (who started process)

Effective user ID (EUID) --

from set user ID bit of file being executed or due to sys call

Saved user ID (SUID) --

place to save the previous UID if one temporarily changes it

Also SGID, EGID, etc..

Executable files have 3 setuid bits

- Setuid bit – set EUID of process to owner's ID
- Setgid bit – set EGID of process to group's ID
- sticky bit:
 - 0 means user with write on directory can rename/remove file
 - 1 means only file owner, directory owner, root can do so

So passwd is a setuid program

program runs at permission level of owner, not user that runs it



```
-r-xr-xr-x 1 root wheel 50512 Feb 10 2011 yes
-r-xr-xr-x 1 root wheel 50832 Feb 10 2011 ypcat
-r-xr-xr-x 1 root wheel 50864 Feb 10 2011 ypmatch
-r-xr-xr-x 1 root wheel 55344 Feb 10 2011 ypwhich
-rwxr-xr-x 2 root wheel 146976 Feb 10 2011 zcat
-rwxr-xr-x 1 root wheel 71 Feb 10 2011 zcmp
-rwxr-xr-x 1 root wheel 4422 Feb 10 2011 zdiff
-rwxr-xr-x 1 root wheel 66 Feb 10 2011 zegrep
-rwxr-xr-x 1 root wheel 66 Feb 10 2011 zfgrep
-rwxr-xr-x 1 root wheel 2017 Feb 10 2011 zforce
-rwxr-xr-x 1 root wheel 4894 Feb 10 2011 zgrep
-rwxr-xr-x 1 root wheel 359968 Feb 10 2011 zip
-rwxr-xr-x 1 root wheel 168432 Feb 10 2011 zipcloak
-rwxr-xr-x 1 root wheel 1188 Feb 10 2011 zipgrep
-rwxr-xr-x 2 root wheel 265392 Feb 10 2011 zipinfo
-rwxr-xr-x 1 root wheel 155440 Feb 10 2011 zipnote
-rwxr-xr-x 1 root wheel 159632 Feb 10 2011 zipsplit
-rwxr-xr-x 1 root wheel 1735 Feb 10 2011 zless
-rwxr-xr-x 1 root wheel 2441 Feb 10 2011 zmore
-rwxr-xr-x 1 root wheel 4954 Feb 10 2011 znew
-r-xr-xr-x 1 root wheel 63424 Apr 29 17:30 zprint
```

```
rist@seclab-laptop1:/usr/bin$ ls -al passwd
```

```
-r-sr-xr-x 1 root wheel 111968 Apr 29 17:30 passwd
```

```
rist@seclab-laptop1:/usr/bin$ █
```

seteuid system call

seteuid can:

- go to SUID or RUID always
- any ID if EUID is 0

```
uid = getuid();
```

```
eid = seteuid();
```

```
seteuid(uid);    // Drop privileges
```

```
...
```

```
seteuid(eid);    // Raise privileges
```

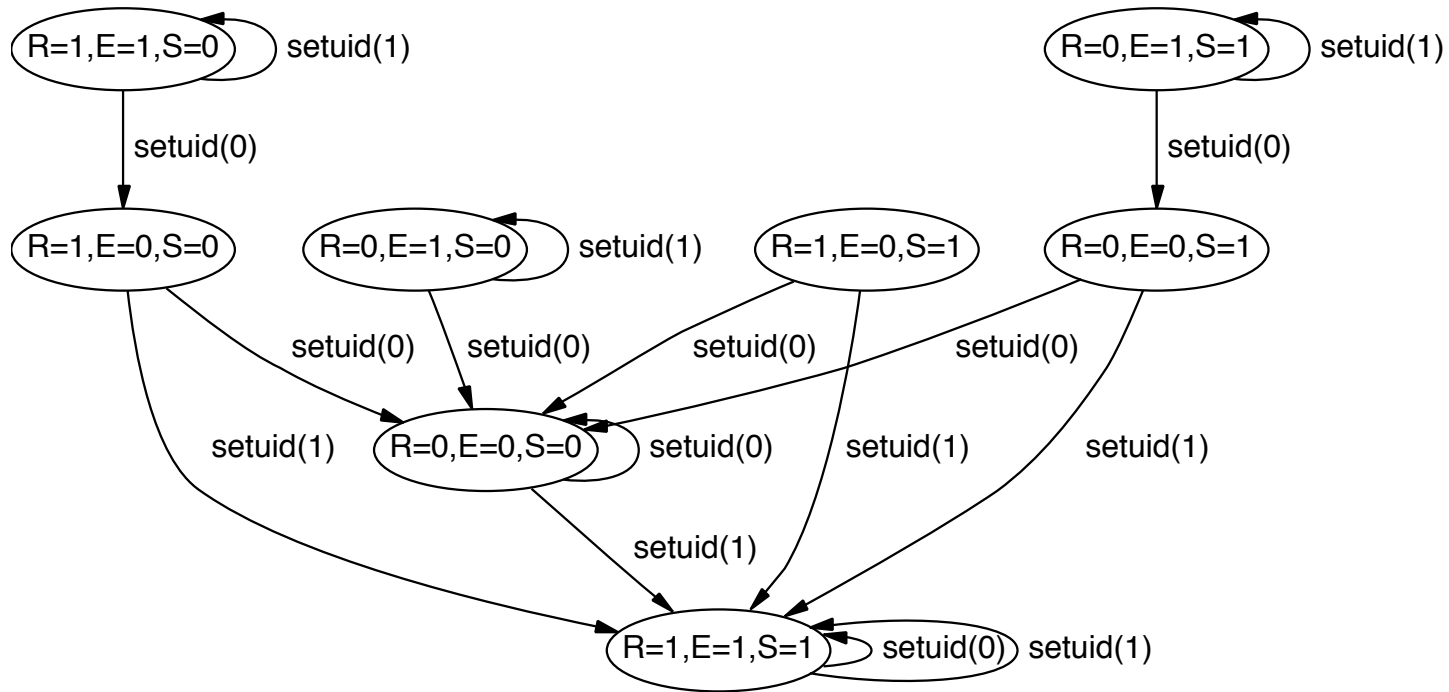
```
file = fopen( "/etc/shadow", "w" );
```

```
...
```

```
seteuid(uid);    // drop privileges
```

Details of setuid more complicated

Chen, Wagner, Dean “Setuid Demystified”



(a) An FSA describing *setuid* in Linux 2.4.18

Setuid allows necessarily privilege escalation but...

- Source of many privilege escalation vulnerabilities

Control-flow hijacking vulnerability (next lecture) in local setuid program gives privilege escalation

Race conditions

Race conditions

Time-of-check-to-time-of-use (TOCTTOU)

```
if( access("/tmp/myfile", R_OK) != 0 ) {  
    exit(-1);  
}  
  
file = open( "/tmp/myfile", "r" );  
read( file, buf, 100 );  
close( file );  
print( "%s\n", buf );
```

Say program is setuid root:
access checks RUID, but open only checks EUID

```
access("/tmp/myfile", R_OK)
```



```
In -s /home/root/.ssh/id_rsa /tmp/myfile
```

```
open( "/tmp/myfile", "r" );
```

```
print( "%s\n", buf );
```

Prints out the root's
secret key...

Better code

```
eid = geteuid();  
ruid = getuid();  
seteuid(ruid);          // drop privileges  
file = open( "/tmp/myfile", "r" );  
read( file, buf, 100 );  
close( file );  
print( "%s\n", buf );
```

Summary

- Multics: seminal multi-user operating system
 - many security features
 - significant auditing performed, achieved high security certifications
- MLS security principles
 - covert channels
- Access controls (matrices, ACLs, capabilities)
- UNIX style file and process permissions