4. OS Protection Mechanisms
ENEE 657

Prof. Tudor Dumitraș
Assistant Professor, ECE
University of Maryland, College Park

http://ter.ps/enee657

Today's Lecture

- Where we've been
  - Memory corruption exploits
  - Cryptography
- Where we're going today
  - Separation of Privileges
  - Confinement
  - Implementation of OS protection mechanisms
- Where we're going next
  - Next week: Empirical security

2

A Note on Pilot Projects

- 2-week project to get initial results and demonstrate feasibility
- Focus on a question that you would like answered
  - For your research, out of curiosity …
  - Some ideas are available on Piazza
- **Post concise (2-3 paragraphs) proposal** on Piazza
  - Problem statement
  - Approach considered for tackling the problem
    - Must describe **concrete tasks**, not vague directions
    - Must **demonstrate that you've thought about the first steps**, and you are not simply paraphrasing the project idea
  - **Deadline: one week from today**

3

Principle of Least Privilege

- What's a privilege?
  - Ability to access or modify a resource
- System has multiple users
  - And multiple components (more on in a bit)
- **Principle of Least Privilege**
  - A user should only have the minimal privileges needed to do his/her work
  - Same for system components

OS Security Model

- Isolation between processes
  - Each process has a user (UID)
    - Two processes with same UID have same permissions
  - A process may access files, network sockets, ….
    - Permission granted according to UID
- Access control matrix [Lampson]

Resources

Principals

	File 1	File 2	File 3	…	File n
User 1	read	write	-	-	read
User 2	write	write	write	-	-
User 3	-	-	-	read	read
…					
User m	read	write	read	write	read

Implementation Requirements

Key component: **reference monitor**

- **Mediates requests** from applications
  - Implements protection policy
  - Enforces isolation and confinement
- Must **always** be invoked:
  - Every application request must be mediated
- **Tamperproof**:
  - Reference monitor cannot be killed
  - … or if killed, then monitored process is killed too
- **Small enough** to be analyzed and validated

16

Implementation Concept #1: Access Control Lists

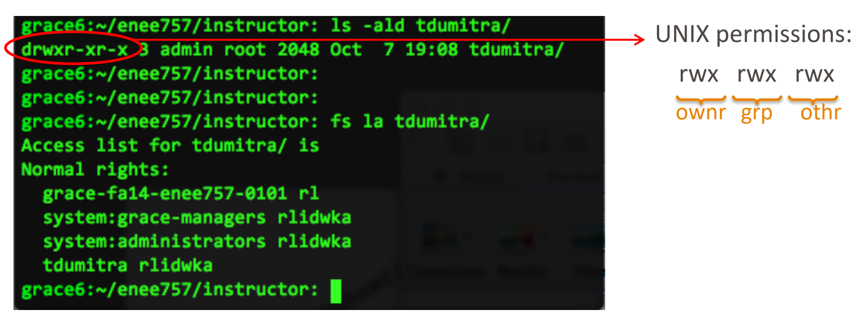
- Access control list (**ACL**)
  - Store column of matrix with resource
  - Relies on authentication: need to know user
  - Delegation: let other process act under current user
    - UNIX su/sudo, Windows UAC

	File 1	File 2	…
User 1	read	write	-
User 2	write	write	-
User 3	-	-	read
…			
User m	Read	write	write

ACL: store in filesystem metadata

17

UNIX Access Control Lists



- UNIX permissions are designed for a single host that manages a local filesystem
  - UIDs: local users
  - Reference monitor: OS kernel

18

AFS Access Control Lists

```
grace6:~/enee757/instructor: ls -ald tdumitra/
drwxr-xr-x 3 admin root 2048 Oct  7 19:08 tdumitra/
grace6:~/enee757/instructor:
grace6:~/enee757/instructor:
grace6:~/enee757/instructor: fs la tdumitra/
Access list for tdumitra/ is
Normal rights:
  grace-fa14-enee757-0101 rl
  system:grace-managers rlidwka
  system:administrators rlidwka
  tdumitra rlidwka
grace6:~/enee757/instructor: 
```

AFS permissions

- The Andrew File System (AFS) is a distributed filesystem
  - Precursor to cloud storage systems
  - Users divided into realms (e.g. UMD, CMU)
  - Reference monitor: file server

Set-id Bits on Executable Unix File

- Three set-id bits
  - Setuid – set EUID of process to ID of file owner
  - Setgid – set EGID of process to GID of file
  - Sticky
    - Off: if user has write permission on directory, can rename or remove files, even if not owner
    - On: only file owner, directory owner, and root can rename or remove file in the directory
- Why needed?

```
grace1:~/enee757: ls -al /usr/bin/passwd
-rwsr-xr-x. 1 root root 30768 Feb 17  2012 /usr/bin/passwd
grace1:~/enee757: ls -al /etc/passwd
-r--r--r-- 1 root root 3521596 Sep  4 18:24 /etc/passwd
```

The Confused Deputy Problem

- Say I want to write a script for students to submit assignments
  - `submit` is invoked by students, compiles and runs tests on the assignment, and places the results in a folder that I can read

```
grace1:~/enee757: ls
instructor/
submit/student1
submit/student2
```

→ My folder (no student access)

→ Students can write

- Say I also want the script to maintain a log file, for debugging
  - `submit` runs with the student's access control permissions
  - Different students cannot access each others' submissions
  - I want to keep the log in the `instructor/` folder
  - How can `submit` update the log file?

21

The Confused Deputy Problem – cont'd

[Hardy, 1988]

- I could make `submit` setuid-instructor
  - At runtime, the script acquires the permissions to write in `instructor/`
  - `submit` can update the logfile
    - Students are still unable to access files in `instructor/` directly
  - Can you see a problem with this?
- `submit` compiles and executes programs that students wrote!
  - A student may submit a program that modifies files in `instructor/` (say, the grade records)
    - Or exploit a vulnerability in my `submit` program to execute code
- The problem is that setuid grants access to all the files I can write (ambient authority)
  - I only wanted to grant write access to the log file
  - But this cannot be expressed in the ACL model!

22

Implementation Concept #2: Capabilities

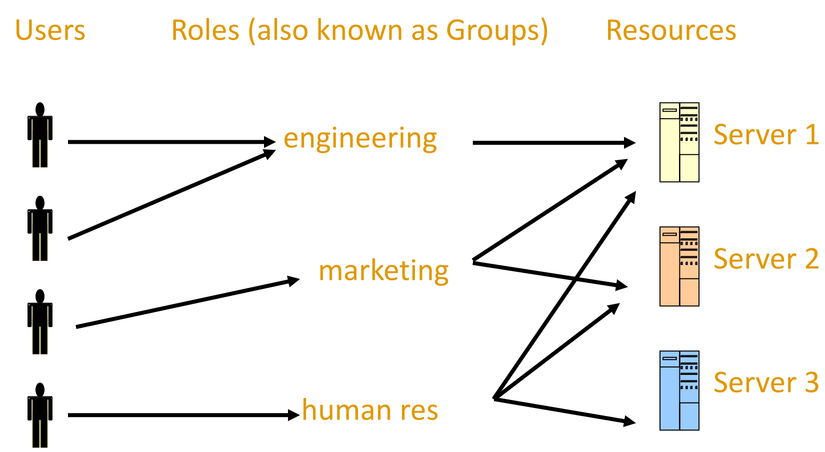
- **Capabilities**
  - User holds a **ticket** for each resource
  - Two variations
    - Store row of matrix with user, under OS control
    - Unforgeable ticket in user space
  - Reference monitor checks ticket: does not need to know identify of user/process
  - Delegation: Process can pass capability at run time

Capability: give user unforgeable ticket

	File 1	File 2	…
User 1	read	write	-
User 2	write	write	-
User 3	-	-	read
…			
User m	Read	write	write

23

Role-Based Access Control



- Role examples: Administrator, PowerUser, User, Guest
  - Assign permissions to roles; each user gets permission
  - Advantage: users change more frequently than roles

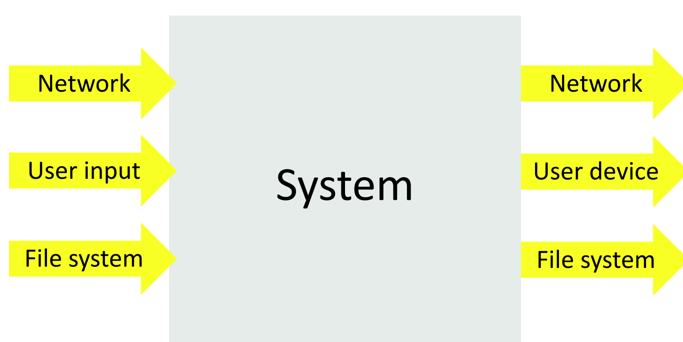
The Confinement Principle

- We've talked about file access control
  - What about other resources?

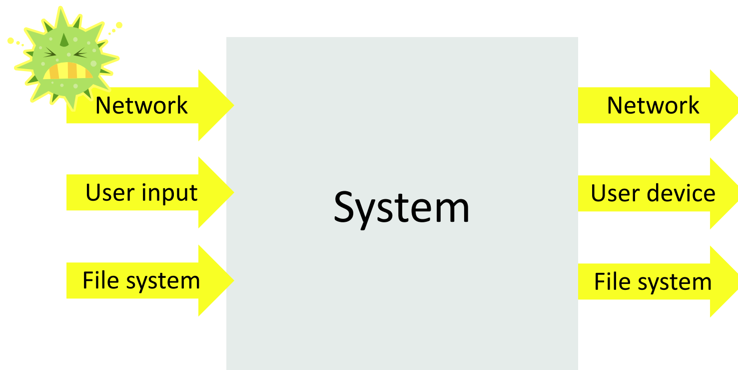
- We often need to run buggy/unstrusted code:
  - programs from untrusted Internet sites:
    - apps, extensions, plug-ins, codecs for media player
  - exposed applications: pdf viewers, outlook
  - legacy daemons: sendmail, bind
  - honeypots

Goal: if application "misbehaves" ⇒ kill it

Monolithic Design

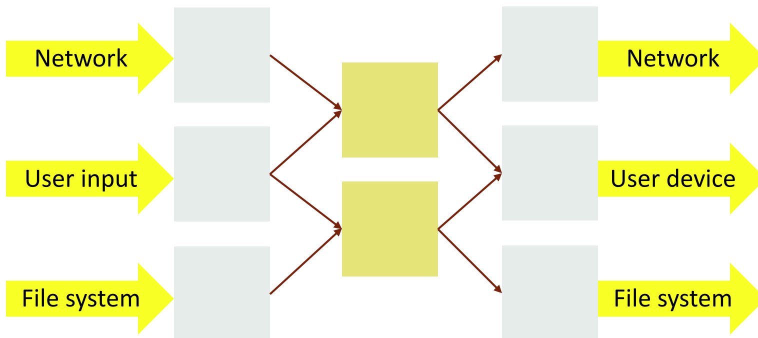


Monolithic Design

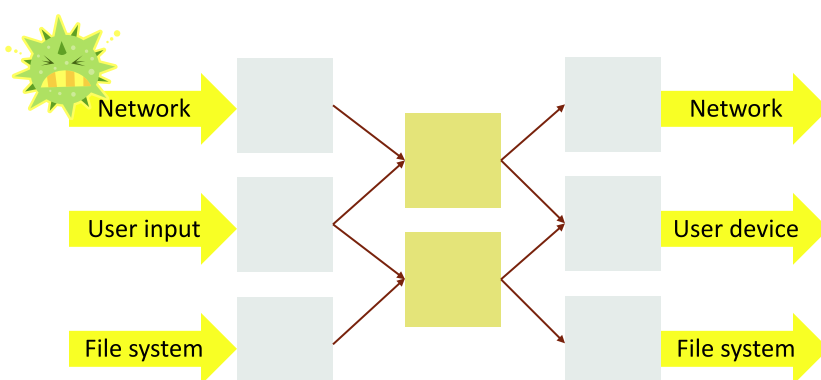


Monolithic Design

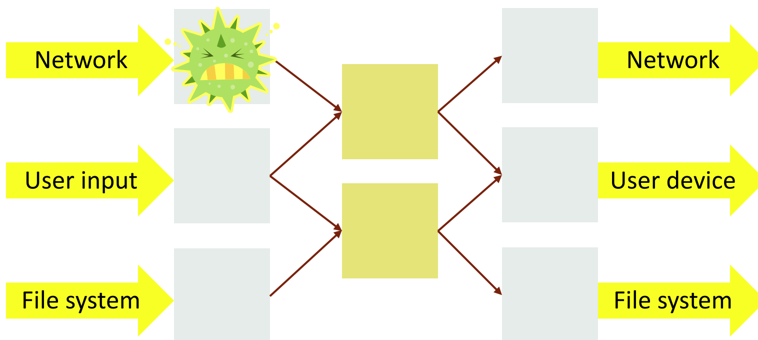
Component Design



Component Design



Component Design

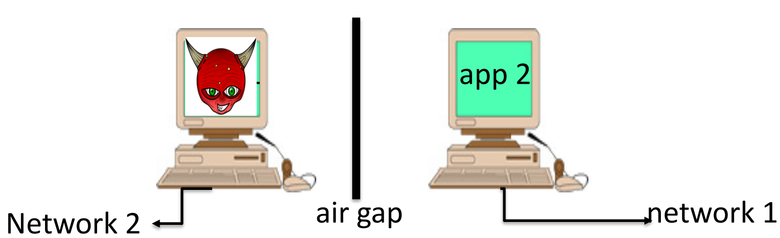


Implementing Confinement

Confinement: ensure misbehaving app cannot harm rest of system

Can be implemented at many levels:

– **Hardware**: run application on isolated hw (air gap)

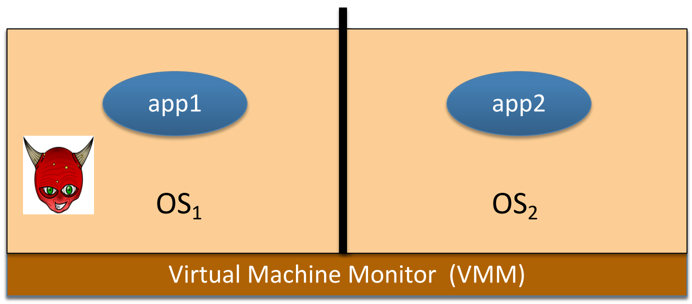


Implementing Confinement

Confinement: ensure misbehaving app cannot harm rest of system

Can be implemented at many levels:

– **Virtual machines**: isolate OS's on a single machine



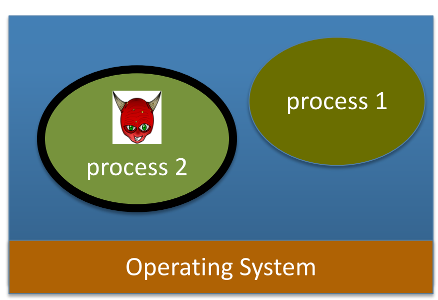
Implementing Confinement

Confinement: ensure misbehaving app cannot harm rest of system

Can be implemented at many levels:

– **Process:** System Call Interposition

Isolate a process in a single operating system

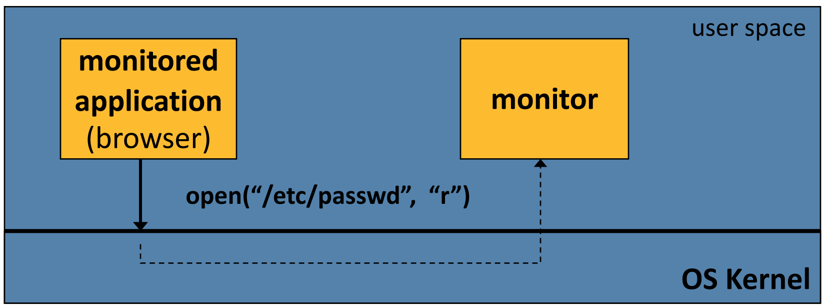


System Call Interposition
[Goldberg+, USENIX Security'96]

- Goal: monitor sys calls and block unauthorized calls
- Implemented with Linux **ptrace**: process tracing

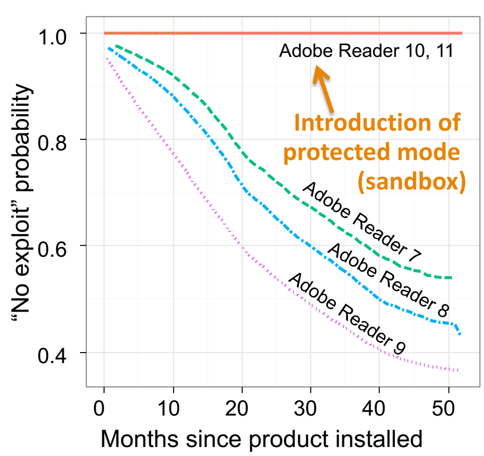
process calls: **ptrace (… , pid_t pid , …)**

and wakes up when **pid** makes sys call



Challenge: how to establish policy for which calls to block?

Measuring Impact of Confinement
[Nayak+, RAID 2014]



37

Confinement: Summary

- Many sandboxing techniques:

  Physical air gap, Virtual air gap (VMMs),

  System call interposition, Software Fault isolation

  Application specific (e.g. Javascript in browser)

- Often complete isolation is inappropriate
  - Apps need to communicate through regulated interfaces

- Hardest aspects of sandboxing:
  - Specifying policy: what can apps do and not do
  - Preventing covert channels

Review of Lecture

- What did we learn?
  - Principals, reference monitor, principle of least privilege
  - ACLs, capabilities, confused deputy
  - Sandboxing
  - Statistical inference

- Sources
  - Dan Boneh, John Mitchell, Vitaly Shmatikov

- What's next?
  - Empirical security
  - Reading: *Setuid Demystified*

39