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
  - **Pilot project proposals due today!**
- Where we're going next
  - Security analytics lab
  - Next week: Network security basics

Pilot Project Proposals

- Due **today at midnight**
  - **Post proposal** on the **Piazza** discussion board
  - Some ideas available on the class Web page
- Proposal should be concise (**2-3 paragraphs**)
  - 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 ideas I gave you

Goals of Security Mechanisms

- Eliminate an **entire class** of attacks
  - Example: harvesting credit card numbers by sniffing network packets used to be common in the '90s. HTTPS stopped that.
  - Challenges:
    - **Arms race**: adversaries find new attacks (e.g., harvesting credit card numbers by hacking point-of-sale systems)
    - Mechanism may not address the **capabilities of real-world adversaries** (we've seen: attacking crypto without breaking the math)
- Make it **less likely** for an attack to succeed
  - Increases the attacker's **work factor**
  - Requires understanding attack techniques
- **Distinguish** between benign and malicious behavior
  - Increasingly using **statistical techniques**

Statistical Inference

- You must understand how to interpret data correctly
- Statistical inference: Methods for drawing conclusions about a population from sample data
- Two key methods
  - Confidence intervals
  - Hypothesis tests (significance tests)

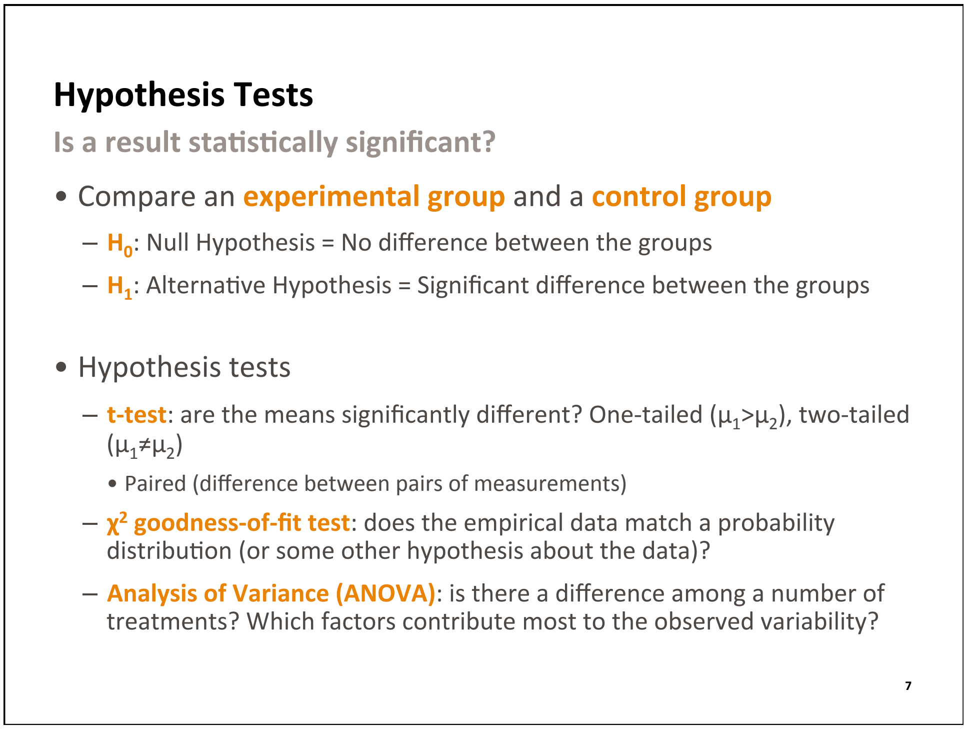
Confidence Intervals

What is the range of likely values?

- 95% confidence interval for the sample mean
  - If we repeated the experiment 100 times, we expect that this interval would include the mean 95/100 times
  - $CI = \mu \pm 1.96 \frac{\sigma}{\sqrt{n}}$

μ: mean
σ: standard deviation
n: number of elements

- Why 95%?
  - No good reason, but widely used
- You can compute confidence intervals for many statistical measures
  - Variance, slope of regression line, effect size, etc.

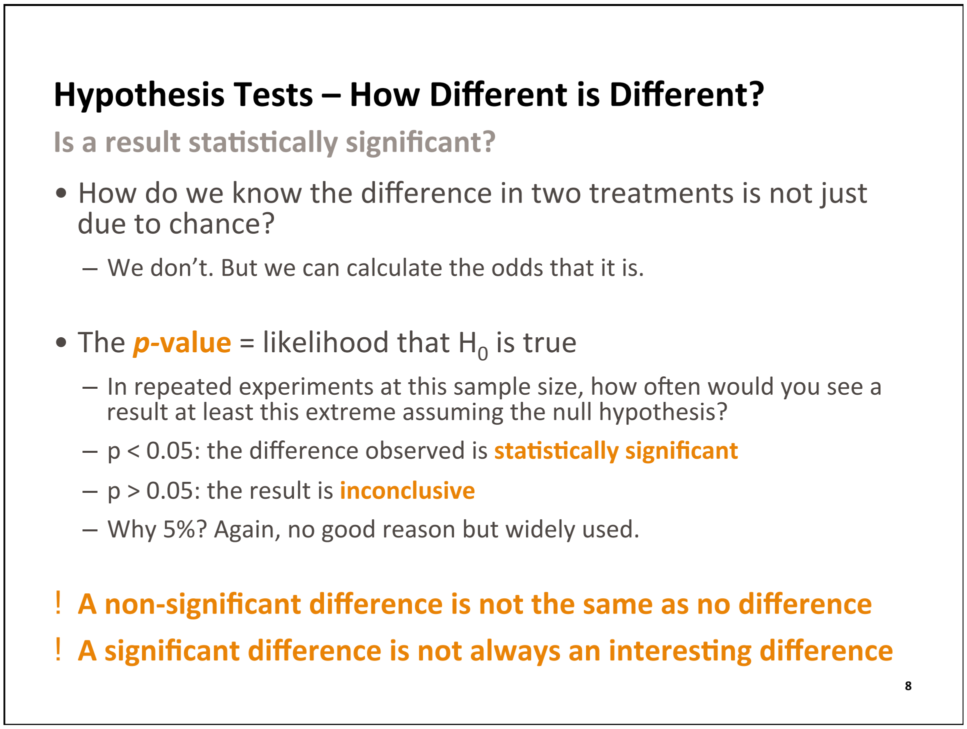


Hypothesis Tests

Is a result statistically significant?

- Compare an **experimental group** and a **control group**
  - **H_0**: Null Hypothesis = No difference between the groups
  - **H_1**: Alternative Hypothesis = Significant difference between the groups

- Hypothesis tests
  - **t-test**: are the means significantly different? One-tailed ($\mu_1 > \mu_2$), two-tailed ($\mu_1 \neq \mu_2$)
    - Paired (difference between pairs of measurements)
  - **χ^2 goodness-of-fit test**: does the empirical data match a probability distribution (or some other hypothesis about the data)?
  - **Analysis of Variance (ANOVA)**: is there a difference among a number of treatments? Which factors contribute most to the observed variability?



Hypothesis Tests – How Different is Different?

Is a result statistically significant?

- How do we know the difference in two treatments is not just due to chance?
  - We don't. But we can calculate the odds that it is.

- The ***p*-value** = likelihood that H_0 is true
  - In repeated experiments at this sample size, how often would you see a result at least this extreme assuming the null hypothesis?
  - $p < 0.05$: the difference observed is **statistically significant**
  - $p > 0.05$: the result is **inconclusive**
  - Why 5%? Again, no good reason but widely used.

! **A non-significant difference is not the same as no difference**

! **A significant difference is not always an interesting difference**

Confusion Matrix

How to determine if your attack detector does a good job?

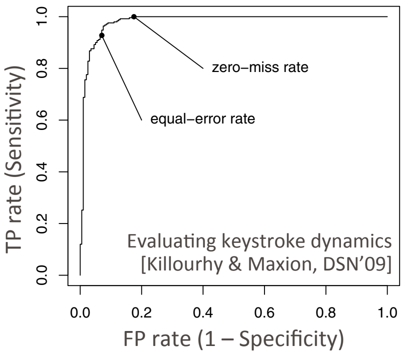
- You need a training set (ground truth) and a testing set
  - Or you can split your ground truth into two data sets
  - Even better: K-fold cross-validation
    - Select K samples without replacement and train classifier multiple times
- You can make a mistake in two different ways

	True -	**True +**
Predicted -	True Negative (TN) *Correct decision*	**False Negative** (FN) *Type 2 error*
Predicted +	**False Positive** (FP) *Type 1 error*	True Positive (TP) *Correct decision*

Evaluating Results

Is it better to have low FPs or low FNs?

- There is usually a **trade-off** between FPs and FNs
  - Reducing type 1 errors causes more type 2 errors, and vice-versa
- **Sensitivity** = TP / (TP+FN)
  - Ability to identify true positives
  - Also called true positive rate
- **Specificity** = TN / (FP + TN)
  - Ability to rule out true negatives
  - Also called true negative rate



Evaluating keystroke dynamics [Killourhy & Maxion, DSN'09]

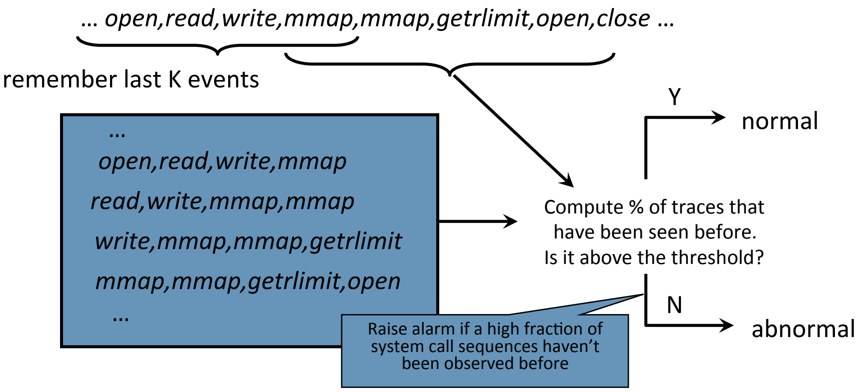
Detecting Attacks and Intrusions

- Observation: to damage host system (e.g. persistent changes) app must make **system calls**:
  - To delete/overwrite files: unlink, open, write
  - To do network attacks: socket, bind, connect, send
- Idea: monitor all system calls and block those that violate **security policy**
  - Language-level: Java runtime environment inspects the stack of the function attempting to access a sensitive resource and checks whether it is permitted to do so
  - OS-level: system call wrapper (more on this in a bit)
  - How do you establish the security policy?

Example: "Immunology" Approach

[Forrest et al., IEEE S&P'96]

- Normal profile: short **sequences of system calls**
  - Use strace on UNIX
- Compute statistical properties and report **anomalies**
  - More on this later



… *open,read,write,mmap,mmap,getrlimit,open,close* …

remember last K events

…
open,read,write,mmap
read,write,mmap,mmap
write,mmap,mmap,getrlimit
mmap,mmap,getrlimit,open
…

Compute % of traces that have been seen before.
Is it above the threshold?

Raise alarm if a high fraction of system call sequences haven't been observed before

Goals of Security Mechanisms

- Eliminate an **entire class** of attacks
  - Example: harvesting credit card numbers by sniffing network packets used to be common in the ’90s. HTTPS stopped that.
  - Challenges:
    - **Arms race**: adversaries find new attacks
      (e.g., harvesting credit card numbers by hacking point-of-sale systems)
    - Mechanism may not address the **capabilities of real-world adversaries**
      (we’ve seen: attacking crypto without breaking the math)
- Make it **less likely** for an attack to succeed
  - Increases the attacker’s **work factor**
  - Requires understanding attack techniques
- **Distinguish** between benign and malicious behavior
  - Increasingly using **statistical techniques**

13

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

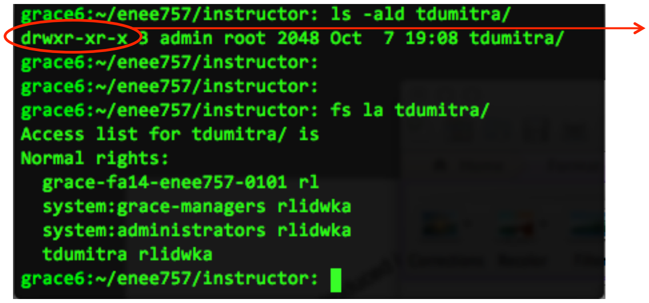
Implementation Concept #1: Access Control Lists

- 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

UNIX Access Control Lists



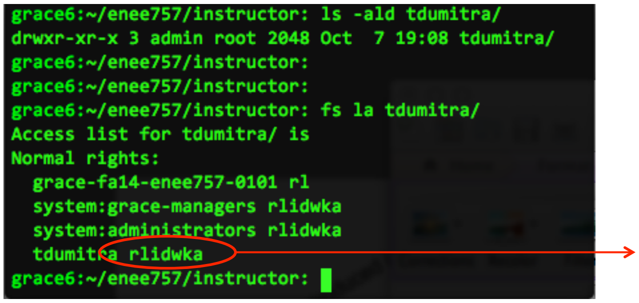
```
grace6:~/enee757/instructor: ls -ald tdumitra/
drwxr-xr-x 3 admin root 2048 Oct  7 19:08 tdumitra/
grace6:~/enee757/instructor:
grace6:~/enee757/instructor:
grace6:~/enee757/instructor: fs la tdumitra/
Access list for tdumitra/ is
Normal rights:
  grace-fa14-enee757-0101 rl
  system:grace-managers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

UNIX permissions:

$\underbrace{\text{rwx}}_{\text{ownr}}\ \underbrace{\text{rwx}}_{\text{grp}}\ \underbrace{\text{rwx}}_{\text{othr}}$

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

AFS Access Control Lists



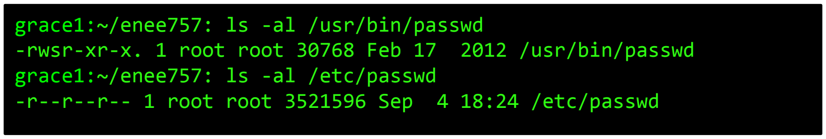
```
grace6:~/enee757/instructor: ls -ald tdumitra/
drwxr-xr-x 3 admin root 2048 Oct  7 19:08 tdumitra/
grace6:~/enee757/instructor:
grace6:~/enee757/instructor:
grace6:~/enee757/instructor: fs la tdumitra/
Access list for tdumitra/ is
Normal rights:
  grace-fa14-enee757-0101 rl
  system:grace-managers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

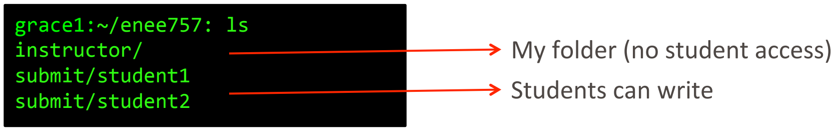
- Three set-id bits
  - Setuid – set EUID of process to ID of file owner
  - Setgid – set EGID of process to GID of file
  - Sticky
    - Off: if user has write permission on directory, can rename or remove files, even if not owner
    - On: only file owner, directory owner, and root can rename or remove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

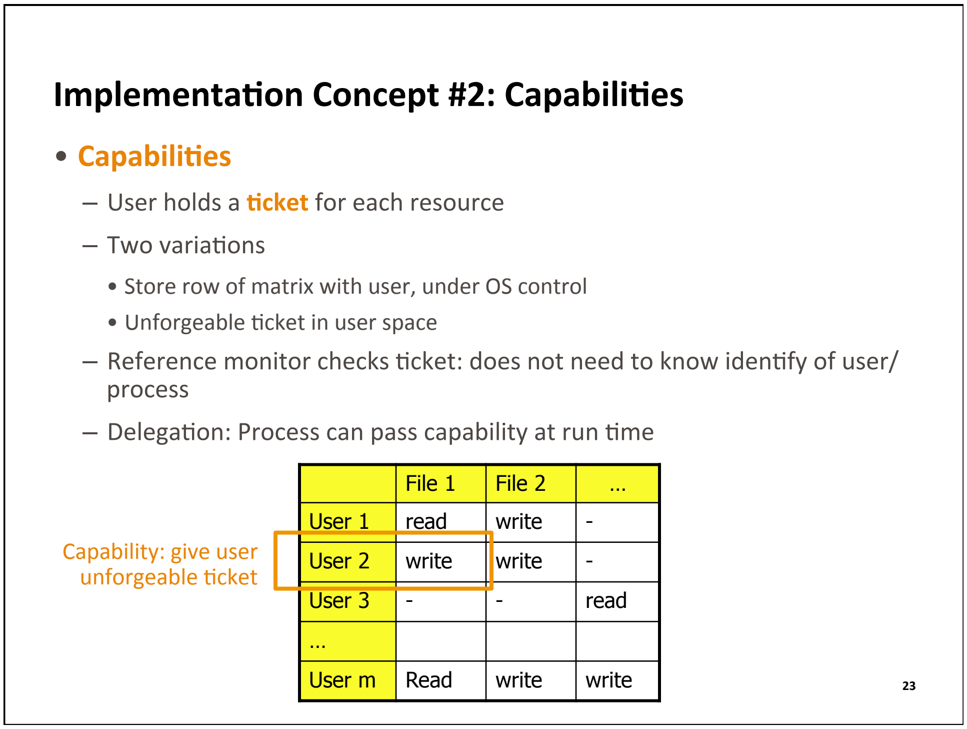
The Confused Deputy Problem – cont'd

[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!



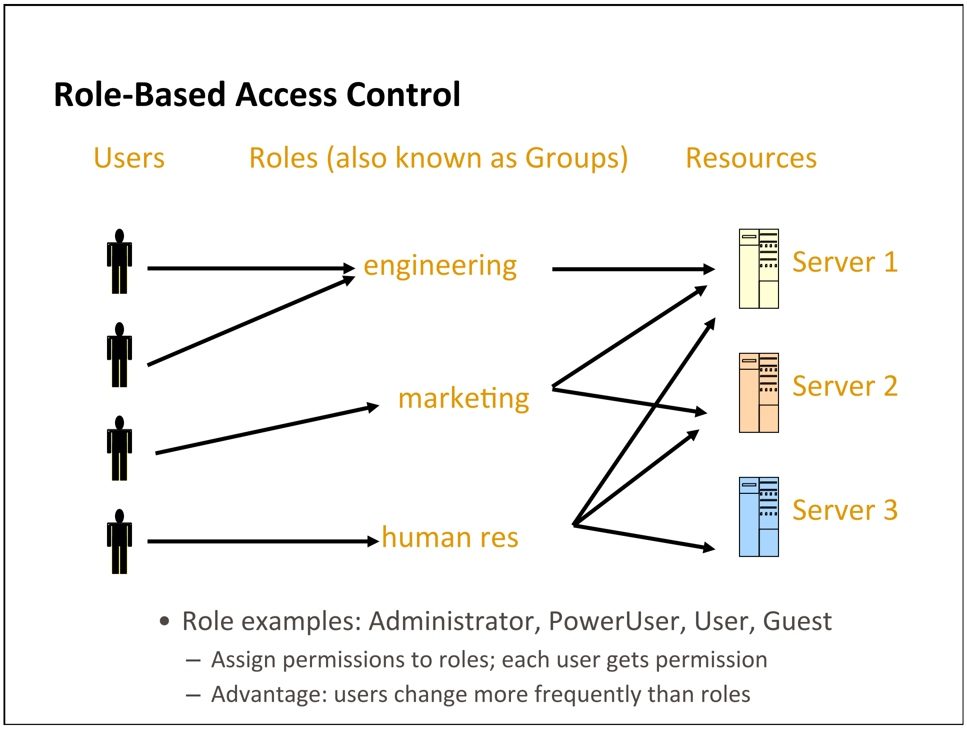
Implementation Concept #2: Capabilities

- **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

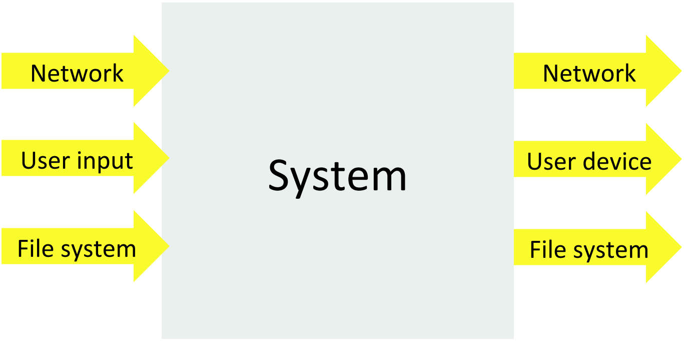
The Confinement Principle

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

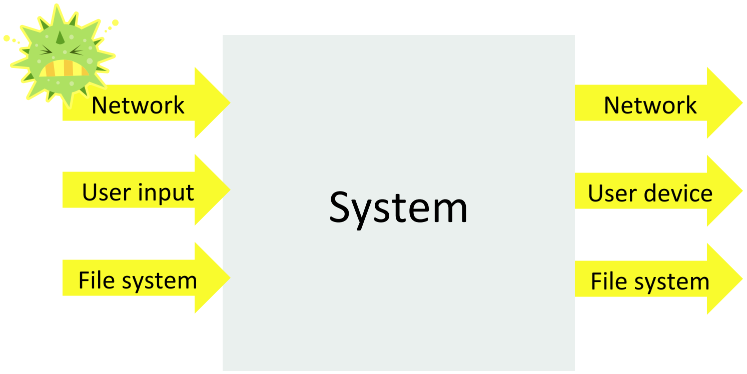
- We often need to run buggy/unstrusted code:
  - programs from untrusted Internet sites:
    - apps, extensions, plug-ins, codecs for media player
  - exposed applications: pdf viewers, outlook
  - legacy daemons: sendmail, bind
  - honeypots

Goal: if application "misbehaves" ⇒ kill it

Monolithic Design



Monolithic Design



Network

User input

File system

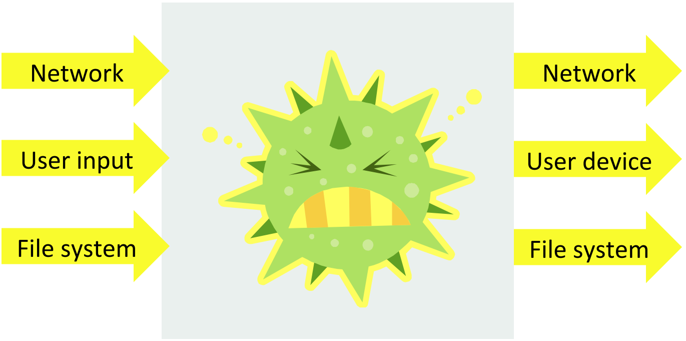
System

Network

User device

File system

Monolithic Design



Network

User input

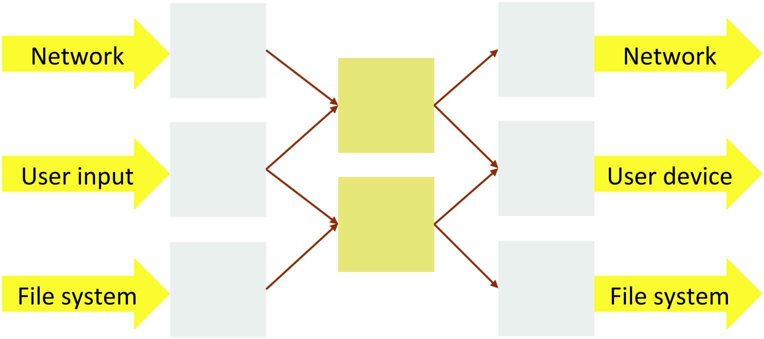
File system

Network

User device

File system

Component Design



Network

User input

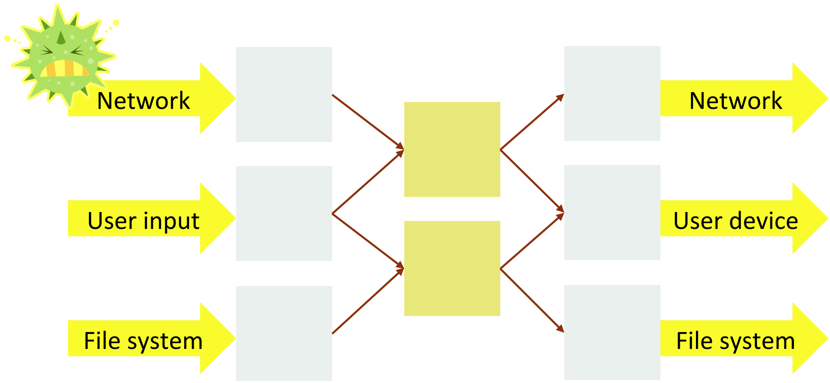
File system

Network

User device

File system

Component Design



Network

User input

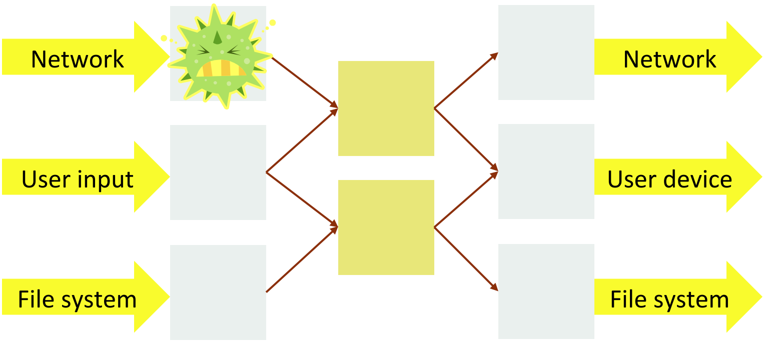
File system

Network

User device

File system

Component Design

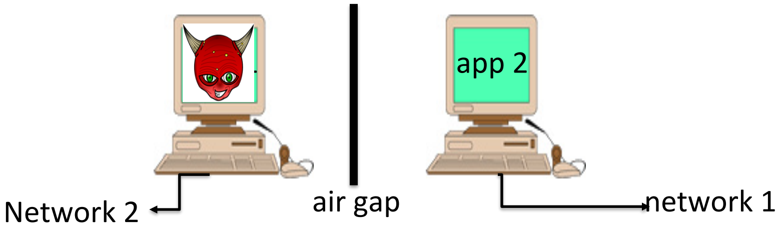


Implementing Confinement

Confinement: ensure misbehaving app cannot harm rest of system

Can be implemented at many levels:

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

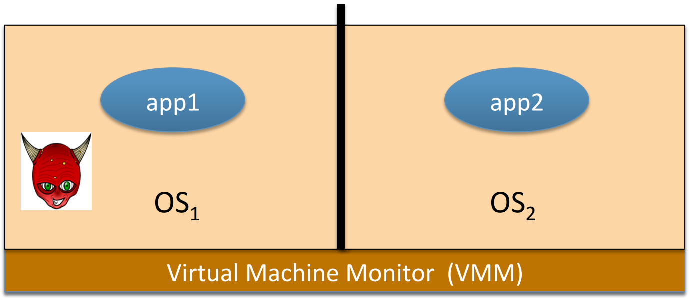


Implementing Confinement

Confinement: ensure misbehaving app cannot harm rest of system

Can be implemented at many levels:

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



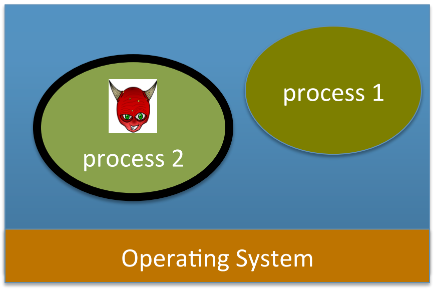
Implementing Confinement

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



Implementing Confinement

Confinement: ensure misbehaving app cannot harm rest of system

Can be implemented at many levels:

- **Threads:** Software Fault Isolation (SFI)
  - Isolating threads sharing same address space

- **Application**: e.g. browser-based confinement

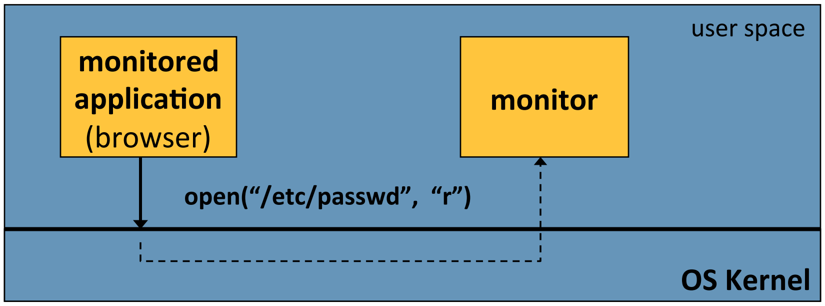
System Call Interposition

[Goldberg+, USENIX Security'96]

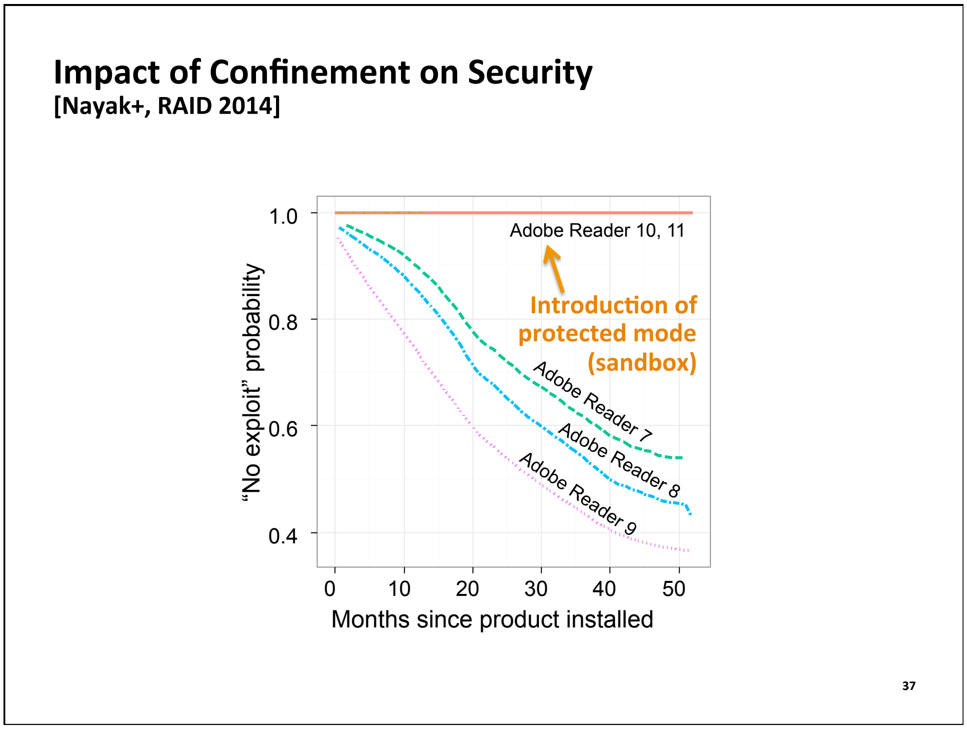
- Goal: monitor sys calls and block unauthorized calls
- Implemented with Linux **ptrace**: process tracing

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

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



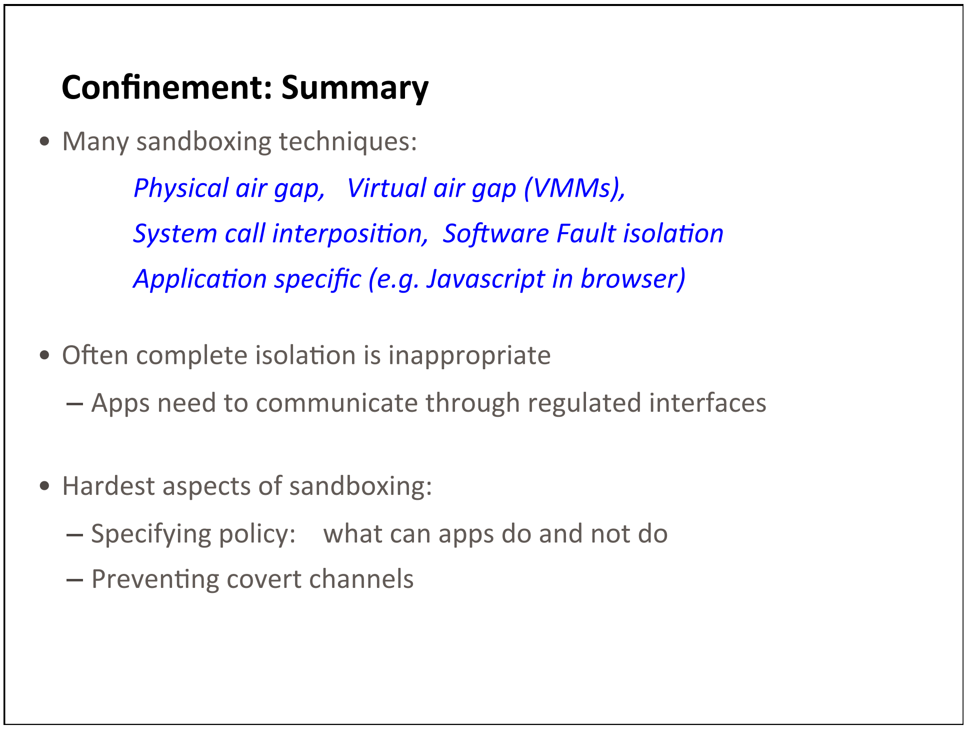
Challenge: how to establish policy for which calls to block?



Impact of Confinement on Security

[Nayak+, RAID 2014]

37



Confinement: Summary

- Many sandboxing techniques:

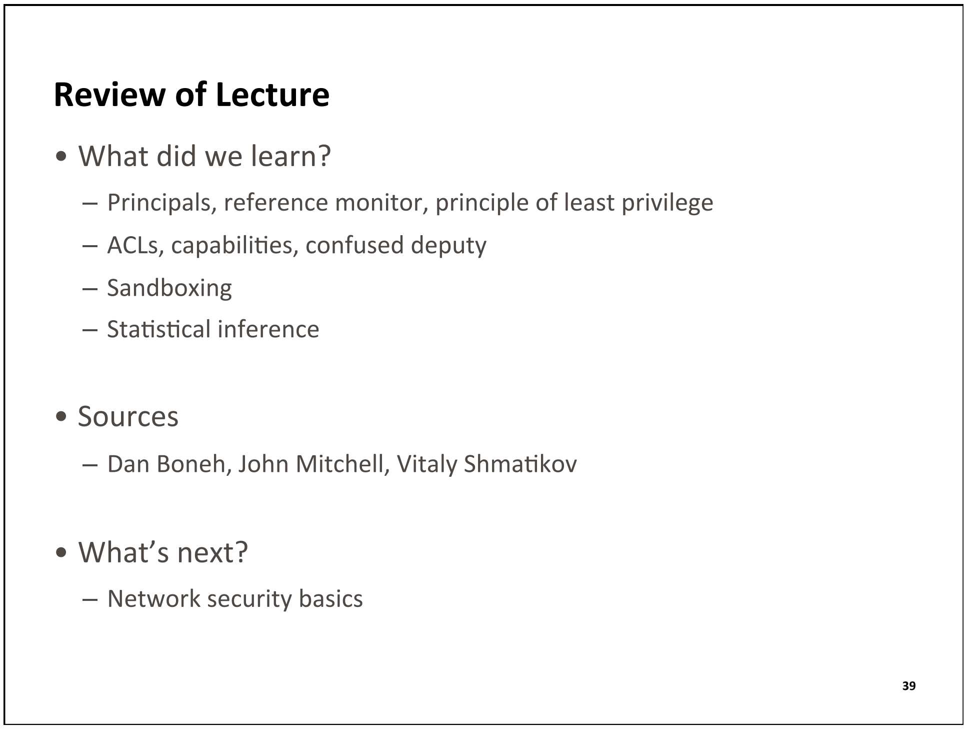
  Physical air gap, Virtual air gap (VMMs),

  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?
  - Network security basics