CSCI 331:
Introduction to Computer Security

Lecture 22:
Privilege separation /
How to give a **good** talk

Instructor: Dan Barowy

Williams

Topics

Finish Info Flow Activity

Principle of Least Privilege

Defense in Depth

Privilege Separation

Giving a Good Talk

Your to-dos

1. Reading response (Thompson), **due Wed 12/8**.
2. Doodle poll (see email), **due NOW**.
3. Final project, **due Friday 12/10 at 5:00pm**.
4. Resubmissions due **Saturday, Dec 18**.
5. Course evaluations **Monday, 12/6**; please bring your laptop.

Secure Information Flow Model

FM = <**N**, **P**, **SC**, {→}, {⊕}>

A set of **objects**.

A set of **principals**.

A set of **security tags**.

A set of **flow relations**.

A set of **join relations**.

Class Activity

FM = <N, P, SC, {→}, {⊕}>

N = { your diary, dinner plans, parents' diary}

P = { you, your little sister (YLS), parents }

SC = { ??? }

→ = { ??? }

⊕ = { ??? }

1. **Only parents** should be able to read **the parents' diary**.
2. **Only you** should be able to read **your diary**.
3. **Anyone** can read the **dinner plans**.

Fill in SC, →, and ⊕ and provide tags **s**.

Class Activity

Your model should be able to answer these questions **mechanically**.

- Can you read your diary?
- Can you write about dinner in your diary?
- Can your parents copy dinner information from their diary into the dinner plans?
- What happens if a page from your diary and a page from your parents diary both just happen to fall out at the same time and stick together. Who can read those pages?

Practicality issues for access controls

Mandatory vs Discretionary Controls

Can parents tell the kids the dinner plans at all?

Not the way we formulated it.

Access controls are **discretionary** in the sense that a principal with a certain access permission is capable of performing that action **unless restrained** by a **mandatory access control**.

Implicit flow

Consider the following program.

Suppose **s**(l) = public and **s**(h) = private.

```
int l;
bool h;
if (h) {
    l = 3
} else {
    l = 42
}
```

The value of h can be deduced because of a “side channel vulnerabilty”.

Side Channel

A **side channel vulnerability** is any vulnerability that exists when **public information** observed during the correct operation of an implementation allows an attacker to **infer and exploit secret state**.

Current Events: Identifying Webpages by Tapping the Electrical Outlet

Shane S. Clark[1] Hossen Mustafa[2] Benjamin Ransford[3]
Jacob Sorber[4] Kevin Fu[5] and Wenyuan Xu**[2,6]

[1]University of Massachusetts Amherst [2]University of South Carolina
[3]University of Washington [4]Clemson University [5]University of Michigan
[6]Zhejiang University

Abstract. Computers plugged into power outlets leak identifiable information by drawing variable amounts of power when performing different tasks. This work examines the extent to which this side channel leaks private information about web browsing to an observer taking measurements at the power outlet. Using direct measurements of AC power consumption with an instrumented outlet, we construct a classifier that correctly identifies unlabeled power traces of webpage activity from a set of 51 candidates with 99% precision and 99% recall. The classifier rejects samples of 441 pages outside the corpus with a false-positive rate of less than 2%. It is also robust to a number of variations in webpage loading conditions, including encryption. When trained on power traces from two computers loading the same webpage, the classifier correctly labels further traces of that webpage from either computer. We identify several reasons for this consistently recognizable power consumption, including system calls, and propose countermeasures to limit the leakage of private information. Characterizing the AC power side channel may help lead to practical countermeasures that protect user privacy from an untrustworthy power infrastructure.

Side Channel Mitigation

Try to ensure that there is no relationship between observable and unobservable state.

An extremely difficult task, especially for programs that *do something*!

Side Channel Mitigation

Alternative: minimize relationship between observable and unobservable state.

```
int l;
bool h;
if (h) {
    l = 3
} else {
    l = 42
}
```

"How many bits of information about `h` does `l` leak?"

`l` leaks 1 bit; since `h` is a 1 bit variable, this is everything an attacker needs.

Further reading: **quantitative information flow**.

Principle of Least Privilege

The **principle of least privilege** means giving a user or process only those privileges which are **essential** to perform its **intended function**.

Defense in Depth

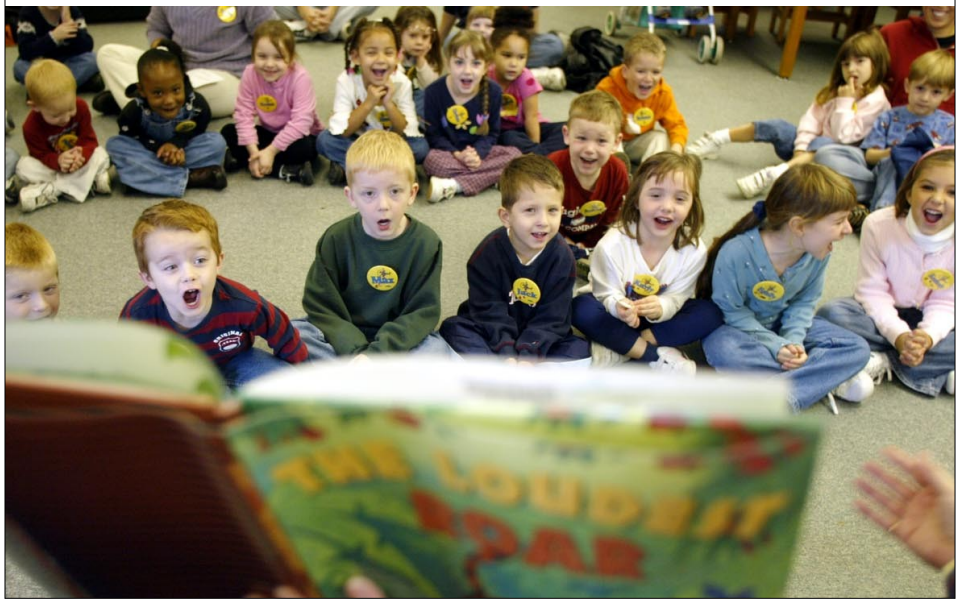
Defense in depth is a technique to enhance overall system security by employing multiple layers of security defenses. Its intent is to provide **redundancy** in the event a security control **fails** or a vulnerability is **exploited**. Different layers can also handle different concerns, such as **personnel**, **procedures**, **technical** or **physical** concerns, as well as considerations relating to the system's **life cycle**.

Paper discussion (Provos)

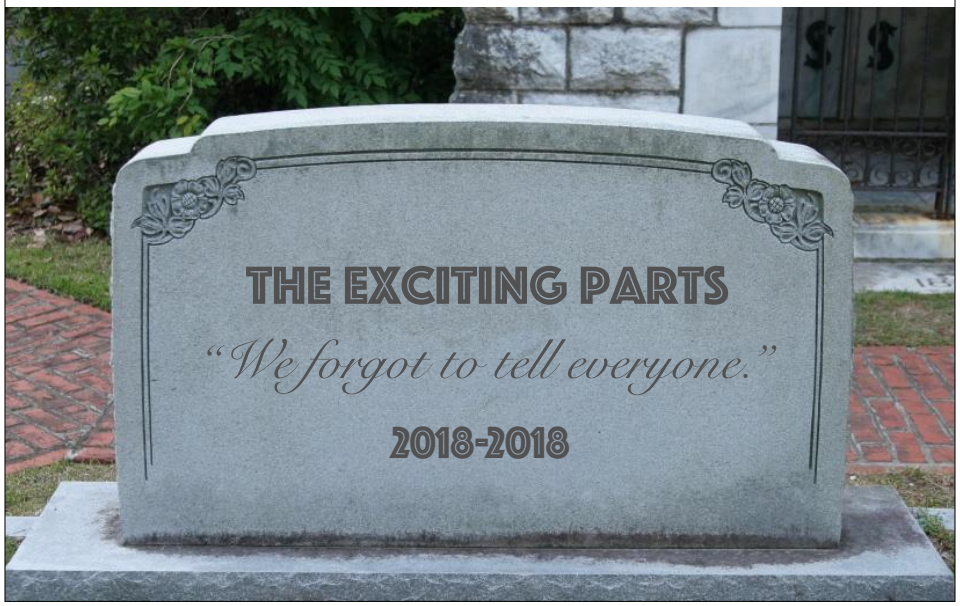
How to give a good talk

Five tips

One: Have a story

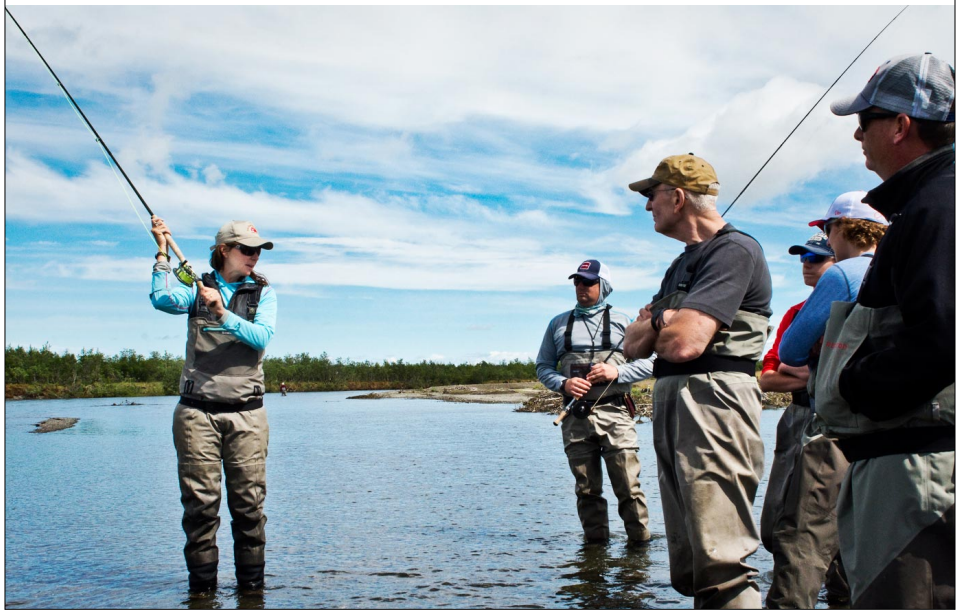


Two: Don't "bury the lede"



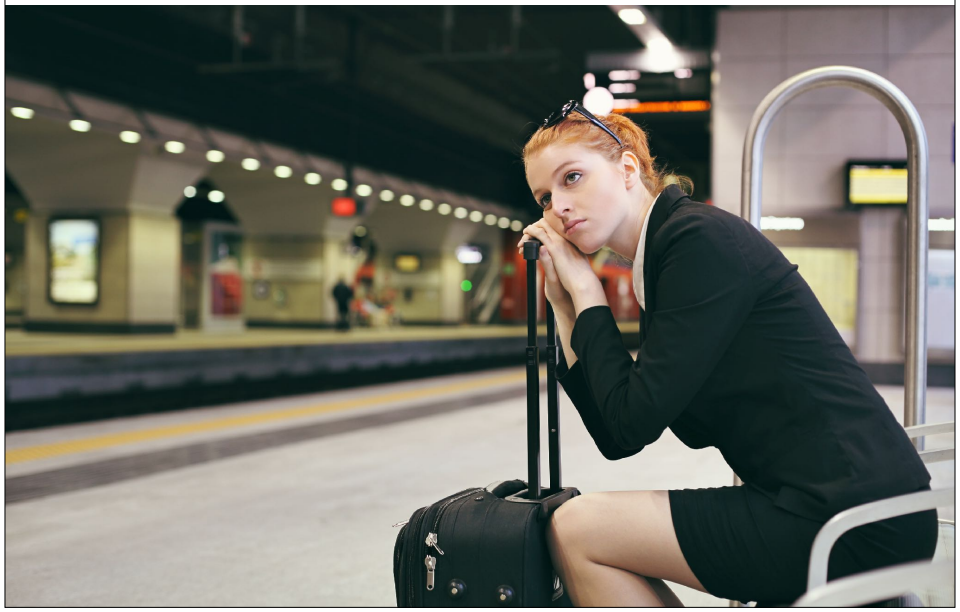
Three: Don't make your audience read

Four: Use examples



Five: Stay on script

Six (oops!): Finish on time



Recap & Next Class

Today we learned:

- Principle of least privilege
- Security in depth
- Privilege separation
- How to give a good talk

Next class:

- Sample talk
- Course evaluations