Introduction to Security

COS 316 Lecture 3

Amit Levy

“Security” is a statement that a system achieves some “goal” despite attacks by “adversaries.”

Goal

What we'd like your system to achieve

- Only Alice can read her files
- Mobile apps should not be send my data to advertisers
- A server should have at least its fair share of the network

Policy

A specification of the goal for the system to follow

- Set file readable only by Alice’s processes (`chmod u+rw go-rwx`)
- Don’t allow outgoing network connections from unprivileged applications
- Allow each network client to send at most 1000 packets per second

Threat Model

Assumptions about what adversaries can do

- Can send arbirtrary data, guess passwords, run arbitrary code, etc
- Physical access?
- Large computing resources?

Mechanism

Software or hardware the system uses to enforce policy

- User accounts, file permissions
- Encryption
- Firewall, packet filters
- Static code analysis

Not *Necessary* for Adversaries to be “Malicious”

- Modeling users or applications as malicious lets us make **strong guarantees** about system behavior.
- Consider programs or users that are not malicious but:
    - Have bugs
    - Difficult to reason about global implications of actions
    - Are not being careful at 4 a.m.

Why is building secure systems hard?

Why is building secure systems hard?

Example: Only the instructors should be able to read the grades database

Why is building secure systems hard?

Example: Only the instructors should be able to read the grades database

Easy to implement positive the aspect of the policy!

```
func canAccessDatabase(user User) (bool) {
  return true
}
```

But security is a *negative* goal. E.g. there shouldn't be no tricky way for a student to get access.

Why is building secure systems hard?

At first glance, the entire software/hardware stack must be involnerable:

- Bug in the database code?
- Convince the operating system to add you to the “instructors” group
- Exploit a bug in the hardware

Why is building secure systems hard?

At first glance, the entire software/hardware stack must be involnerable:

- Bug in the database code?
- Convince the operating system to add you to the "instructors" group
- Exploit a bug in the hardware

Secure systems with one weird trick:

- Design the system such that only a small part enforces security and needs auditing.

What goes wrong #1: Policy

System correctly enforces policy but the policy is inadequate

Example: Fairfax County, VA school system [1]

- **Students** can only access their own files
- **Teachers**
  - Can change user's passwords if a student in their class
  - Can add users as students to their class
- **Superintendent** has access to everyone's files

What goes wrong #1: Policy

System correctly enforces policy but the policy is inadequate

Example: Fairfax County, VA school system [1]

- **Students** can only access their own files
- **Teachers**
    - Can change user's passwords if a student in their class
    - Can add users as students to their class
- **Superintendent** has access to everyone's files

What could go wrong if a student gets a teacher's password? (5 minutes)

More examples

- Sarah Palin’s e-mail hack [2]
- Mat Honan’s accounts at Amazon, Apple, Google, etc.[3]

What goes wrong #2: Threat model

Problems with the threat model's assumptions: designers underestimated an adversary's capabilities

What goes wrong #2: Mechanism

Mechanism can't actually achieve policy or is buggy.

Bugs are a huge problem:

- Rule of thumb: 1 bug per 1000 lines of code
    - PostgreSQL database: > 2M LoC, Linux: > 15M LoC, Firefox: > 20M LoC
- Bug finding/elimination active area of systems research:
    - Static code analysis (e.g. Clang's `scan-build`, BLAST, Coverity, KInt...)
    - Certified software using rich type-systems (Coq, Isabella, Agda) or constraint solving

What goes wrong #2: Mechanism

- Policy:
  - Only users with access to a file can learn it's contents
  - Any program can write non-sensitive data to a shared system log: `/var/log/messages`
- Threat Model: Users can run arbitrary programs, can't break logins
- Mechanism: Each file has read/write for user and "other"
  - Users: `Alice`, `Bob`, `Eve`

What goes wrong #2: Mechanism

- Policy:
  - Only users with access to a file can learn it's contents
  - Any program can write non-sensitive data to a shared system log: `/var/log/messages`
- Threat Model: Users can run arbitrary programs, can't break logins
- Mechanism: Each file has read/write for user and "other"
  - Users: `Alice`, `Bob`, `Eve`

Construct a permissions scheme using the mechanism that enforces policy

Later in the semester we’ll talk about how it’s possible to enforce these kinds of policies using Mandatory Access Control or Information Flow Control.

Up Next

Assignment 1

- Due 9/24 @ 11pm
- Precept *tomorrow* should help... a lot...

Next Time: Introduction to Operating systems

Then we're diving in

References

[1] *http://catless.ncl.ac.uk/Risks/26.02.html#subj7.1.*

[2] *http://en.wikipedia.org/wiki/Sarah_Palin_email_hack.*

[3] *http://www.wired.com/gadgetlab/2012/08/apple-amazon-mat-honan-hacking/all/.*