Computer Security

01r. Recitation: Introduction

Rutgers University

Fall 2019

419 Recitations

TAs

- Fan Zhang
  - fz110@rutgers.edu
- Shuo Zhang
  - sz514@rutgers.edu

- You may not always have the same TA
- You can schedule a meeting with any available TA

Recitation Topics

Recitations will cover:

- Homework review & project guidance
- Extended coverage of course material
- Exam preparation

Key Concepts From Lecture 1

CIA Triad

Triad = group of three things

Model for thinking about computer security

1. **Confidentiality**
   - Restrict access to data and resources (e.g., computing, network) to only those who <u>need to know</u>
   - This access is defined by a <u>policy</u>
   - Requires
     - <u>Identification</u>: who is the user (or computer or application)?
     - <u>Authentication</u>: verify the user (or computer or application)
     - <u>Authorization</u>: check the policy to see if the user is has access
   - **Implemented via access control mechanisms or cryptography**

Authentication establishes the *<u>integrity</u>* of the user

2. **Integrity**

3. **Availability**

CIA Triad

Triad = group of three things

Model for thinking about computer security

1. **Confidentiality**
2. **Integrity**
   - Establish trustworthiness of data, users, and resources
   - Detect tampering
   - Validate (authenticate) the identity of users/systems/services
   - **Implemented via authentication algorithms and/or cryptography**
3. **Availability**

CIA Triad

Triad = group of three things

Model for thinking about computer security

1. **Confidentiality**
2. **Integrity**
3. **Availability**
   - Ensure data and resources are accessible and perform adequately
   - Includes recovery from failure
   - **Implemented via OS resource management, OS thread scheduler, firewalls, load balancers, replication & backups**

Security Engineering

Combination of

1. Policy (rules)
2. Mechanisms (implementation)
3. Assurance (integrity of the mechanisms and policy)
4. Incentives (the human factor)

Engineering = not just the design of the system but understanding the trade-offs (time, money, complexity, features, …)

Policies & Mechanisms

Security Policy: *the rules of what is and is not allowed*

Security Mechanism: *method for enforcing the policy*

- Mechanisms can be procedural or technical

  For example:

  – Procedural: inspect a student's ID card when they submit an exam

  – Technical: an operating system enforces read restrictions to prevent a student from copying another's assignment

- We assume that a security policy is correct and unambiguous

Security Assumptions

- The heart of all security rests on assumptions about:
  - Type of security needed
  - Environment where the system is deployed
  - Trusted components & principals (users, other systems)

> Definition
>
> **Principal**: any entity that can be identified and authenticated – users, computers, processes, services

- Example
  - You need a key to open a locked door
  - You assume that the lock is a trusted component and is secure against lock picking
  - BUT … a skilled lock picker can open the lock

- Your assumptions are wrong *IF*
  - The environment has a skilled, untrustworthy lock picker
  - The lock is trivial to pick (bad mechanism)

Trust: **Trustworthy** components

May have the capabilities to break security policies
… but will not do so: they will follow the policy

Examples

- A *trustworthy* lock picker will not bypass security unless properly authorized
- A *trustworthy* CPU will correctly enforce memory protections and not allow a user to read regions of memory disallowed by the operating system
- A trustworthy operating system will not allow you to read or modify files to which you do not have access permissions

- If a core component turns out to be <u>*not trustworthy*</u> then the security of the ***entire*** system may be in jeopardy
- Example: a malicious boot loader can patch the code of the operating system that, in turn, can run a malicious program or change the behavior of programs

Assurance = our faith in the system

Assurance = how much we can trust a system

This includes

- **Specifications**
  - Statement (formal or informal) of the desired functioning of the system
- **Design**
  - The components that will implement the specification
- **Implementation**
  - The creation of a system that satisfies the design (hence, satisfies the specification)
  - Difficult (impossible) to prove the correctness of the implement of a complex system
- **Testing & auditing**
  - **Auditing** = inspecting the code for security-critical bugs
  - Because we usually cannot prove the correctness of a system, we rely on extensive testing to get that lucky feeling that it works
    - **Functional testing** to assess that the system works as desired
    - Also **penetration testing** to assess that the system follows policy and is resilient to bad inputs, missing components, unexpected events, etc.

Incentives = the human factor

- **Defense**
  - Do we trust employees and the staff responsible for implementing the policy and mechanisms?
  - What does it take to bribe, burglarize, or blackmail them?
  - How important is it to protect your resources?
- **Offense**
  - How motivated are the attackers?
  - Will they be joy hackers, industrial spies, nation states, …?
  - How important is it to attack the target?
- Incentives & motivation (on both sides) determine the engineering tradeoffs in engineering a secure system
  - Balance cost, complexity, time to deploy, perception

Incentives ⇒ Human factors

Security is as weak as its weakest link

- People are often that weakest link
  - It's often easy to **masquerade** as another person if you can steal or guess access credentials (e.g., a user name and password)
  - **Corrupt insiders** are considered trusted and can get through most security measures
  - Untrained people can make **honest mistakes** that compromise security
    - This includes untrained system administrators (bad configurations),
poor programmers (poor implementations),
and employees that abuse privileges ("I didn't know I wasn't allowed to do that")

- Social engineering is a powerful tactic
  - People tend to be trusting and usually try to help … that can backfire
  - Convince someone to give you their credentials or access to the data you need
    - Get ID/password, get someone to let you into the company, impersonate yourself as someone else, …

Risk analysis

- Security is an engineering problem
  - Engineering involves making compromises
    - Cost, time, complexity, convenience, impact on day-to-day life
    - Likelihood of attack and the resulting loss
- Prioritize protection against likely attacks
- Risk is a function of the environment
  - Is a computer accessible over the Internet or only locally?
  - What employees have legitimate access to the system?
  - How likely is it that employees can be bribed?
  - Etc. …
- Risks may change over time
  - New bugs discovered, revised policies, changing network connectivity, new employees, …

Security investment

- Security costs money
  - Physical security: locks, guards, cameras, barbed wire, …
  - Computer security: skilled employees, specifications, design, implementation, penetration testing, protocol validation, etc.
- Security provides no reward
  - Designing a secure system costs extra $ and takes more time
  - It does not make a system faster, easier to use, or operate better
- This impacts the business decision of how much effort to put into security
  - What is the value of the possible loss of reputation, money, intellectual property, etc. if there is a security breach?

Trusted Computing Base (TCB)

- The entire set of components (hardware, firmware, software) that are critical to the security of a system
- The **TCB** implements the mechanisms that implement and enforce the security of a system.
- Any vulnerabilities in the TCB can affect the security of the entire system

The end