Software Security
COSC 466/566
Spring 2023

Dr. Doowon Kim

THE UNIVERSITY OF
TENNESSEE

Schedule

- Next class (Jan 27) is canceled. No class this Friday.

Today’s class

- Security mindset
- What’s the computer security?

Security mindset example

- Any example that you want to share today?

What is computer security?

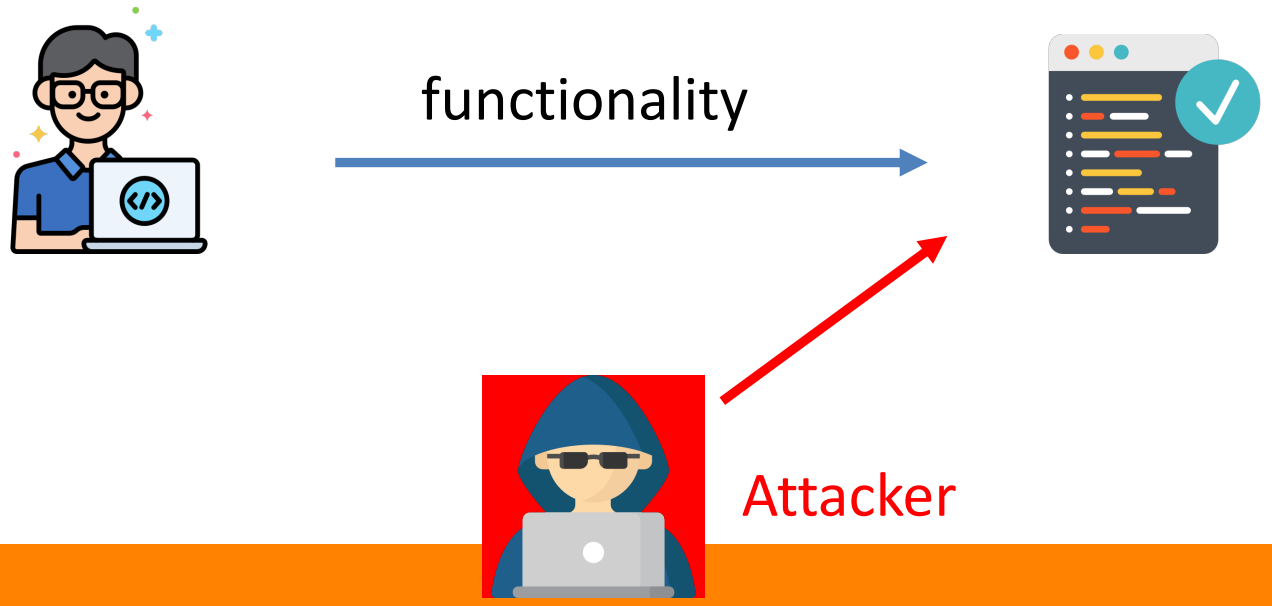
- Normally, we are concerned with functionality and correctness.

- Security is also a form of correctness.

What is computer security?

The key difference:

- Security involves an adversary who is active and malicious.
- Attackers seek to circumvent protective measures.



What's the diff between you and attackers?

- Attackers are not normal users

- Normal users: try to avoid bugs/flaws
- Attackers: try to find the bugs/flaws out and to exploit them

What does it mean to be secure?

- There is no such thing as security, only degrees of insecurity.

- Goal: just raise the bar for the attacker
  - Too difficult
  - Too expensive

- Ultimately, we want to mitigate undesired behavior

Basic Security Principles

- Confidentiality: an attacker cannot *recover protected data*.
  - Information Leak (e.g., side channel)
- Integrity: an attacker cannot *modify protected data*.
  - Data Tampering
- Availability: an attacker cannot *stop/hinder computation*.
  - Denial of Service Attack

- *These fundamental CIA properties are used in different contexts of software security policies and mechanisms*

Security Analysis

- Given a software system, is the system secure?
- *No simple answer, it depends on*
  - What is the interface? (What is the attack surface?)
    - Amazon Echo: Voice
    - Camera (e.g., Self-driving cars): Video
    - …
  - What are the assets? (How profitable is an attack?)
    - Hacking Amazon accounts: Credit cards information, personal info.
    - Hijacking Internet Sessions (e.g., eBay): Can buy things, but they will be shipped to me.
    - …
  - What are the goals? (What drives an attacker?)
    - State-funded attackers (Advanced Persistent Threat)
    - Money (Sell personal information)
    - Leverage compromised machines to launch other attacks (e.g., DDoS)
    - …

Threat Model

- Threat Model: Define an attacker's abilities and resources.
  - Precisely describe who you are dealing with and how capable the bad guys are
  - Example: You have a system that protects a web server
    - An attacker is remotely accessing the web server
    - An administrator of the web server is trusted
    - An attacker may intercept connections between client users and the web server (e.g., Man-in-the-Middle attack)

Cost of Security

- A security system is
  - expensive to develop (man month – X developers for Y months),
  - may incur performance and space overhead,
    - Adding more code will incur performance overhead
    - Changing data structures may incur performance overhead too because it affects cache behaviors of CPU
  - may be inconvenient for users.
    - User Access Control – from Windows Vista
- A practical security solution is
  - less expensive to develop (general solution – resilient to software/system updates, not an ad-hoc patch)
  - low performance and space overhead (less impact on data structures)
  - easy to use.

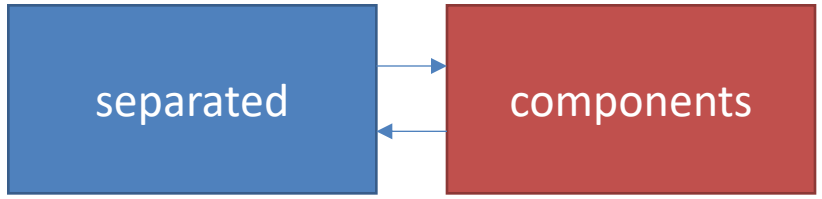
Security concepts

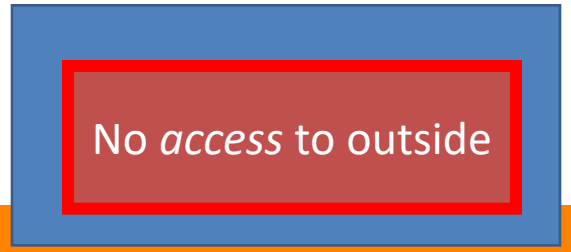
- Least privilege:
  A component (e.g., a process, a user, or a program, depending on the subject) must be able to access only the information and resources that are necessary for its legitimate purpose (i.e., least information is granted for the purpose).

  - Any privilege that is further removed from the component will break the legitimate functionality (purpose).
  - *No additional privilege is needed* for all functionalities of the components.

Security concepts

- Separation
  - Separate individual components into smallest sub-components possible. The sub-components will communicate with each other.
  - Each small component will run with the least privilege.
  - At the boundary of each component, permissions will be checked.
- Isolation vs. Separation
  - Two components are separated: They do not overlap at all. They are completely separated.
  - Isolation does not necessarily mean that they are separated (Error-prone). https://en.wikipedia.org/wiki/Security_and_safety_features_new_to_Windows_Vista#Application_isolation





Security concepts

- Trust
  - Some components are assumed to be trusted
    - Applications trust OS (i.e., kernel)
    - OSes trust hardware
- Correctness
  - *Formal verification* ensures that if a component *correctly* implements a verified specification that can be trusted.
    - Informally, formal verification explores all possible ways including every single corner cases to see whether predefined properties (e.g., does this component leak secret information?) are satisfied.
    - In practice, implementations are often not correct.
    - If you are interested in, learn “Coq: Proof Assistant.”

Security Implementations

- Each process in modern operating systems (e.g., Windows/Linux) has their own memory space and is not supposed to access memory.
  - Enforced by Virtual Memory through hardware:
    1. In x86, each process maintains a page table address on the CR3 register (which is a table of addresses of memory pages that belong to the process).
    2. When the CPU accesses memory, it looks up the table according to the address stored in the CR3 register.
    3. CR3 can be only changed by the kernel.

  - DOS and Windows 3.1 ***DID NOT*** have this luxury – What could go wrong? A memory bug in one application on Windows 3.1 can crash the entire OS.

  - We are using Windows 10: All good? Of course no.
    - CreateRemoteThread – Creating a thread on another process.
    - Library Injection through Window Event Hooking
    - …

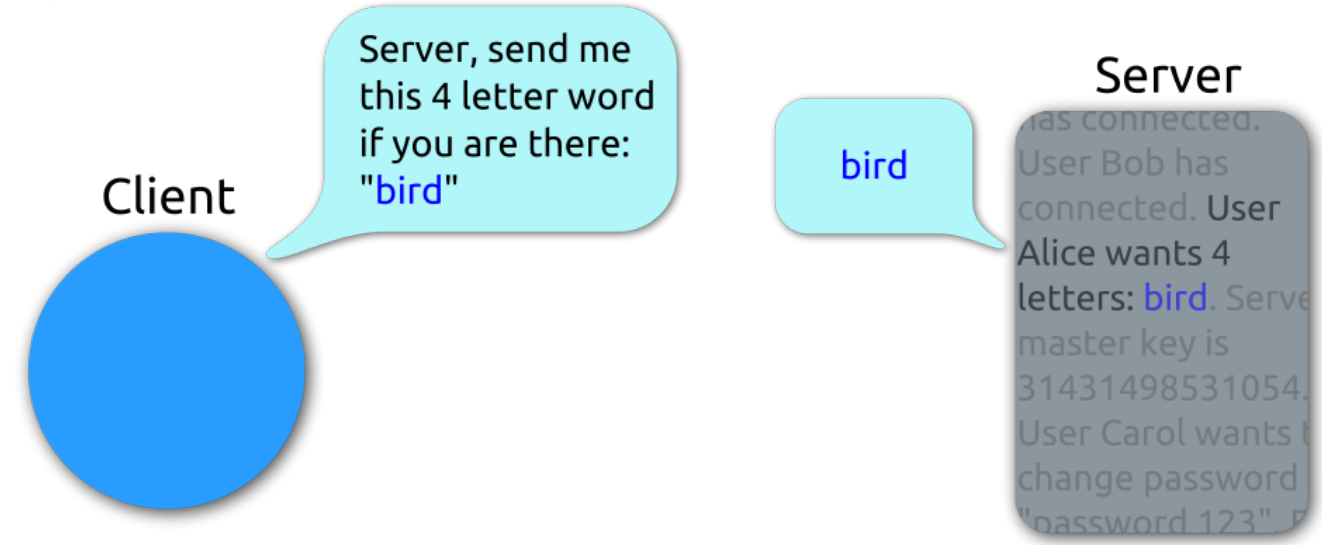
Ex) The Heartbleed Bug

- A security bug in the OpenSSL cryptography library, which is widely used implementation of the Transport Layer Security (TLS) protocol **[wikipedia]**
  - This weakness allows stealing the information
    - user names & passwards
    - instant messgaes & emails
    - business critical documents & communication
    - more..

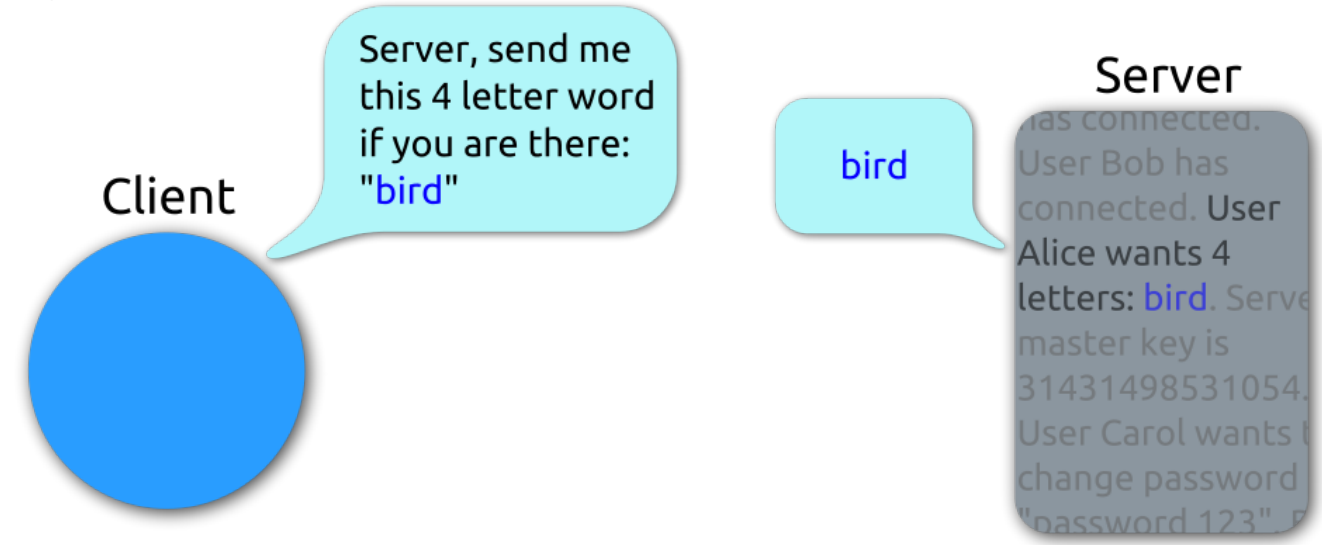
Ex) The Heartbleed Bug

- The vulnerability is a “buffer over-read”
  - Software (accidently) allows more data to be read than should be allowed

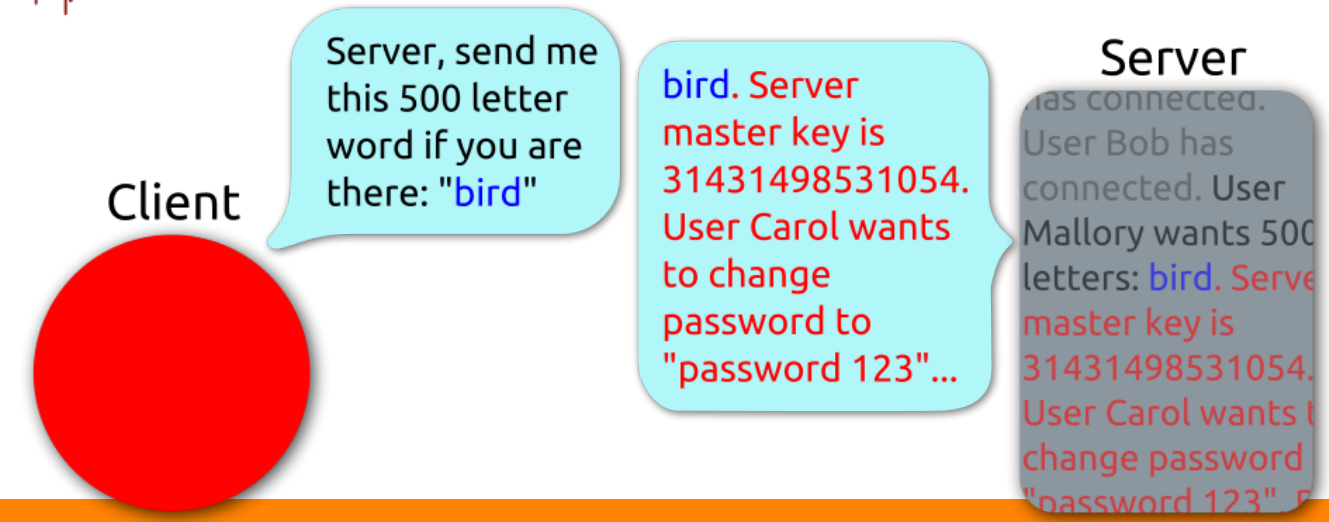
Heartbeat – Normal usage



Heartbeat – Normal usage



Heartbeat – Malicious usage



Ex) The Heartbleed Bug

- The vulnerability is a “buffer over-read”
  - Software (accidently) allows more data to be read than should be allowed
- It is a simple programming bug, but it is hard to discover
  - Vulnerable OpenSSL was released on March 14, 2012
  - Google’s security team reported Heartbleed on April 1, 2014

How to find a problem?

- Manual program inspection
  - Maybe effective
  - But humans are not good at
    - Repetitive and tedious tasks
    - Maintaining large amounts of detail
- Automated program analysis
  - Replace human inspection to find software problems
  - Support inspection by
    - Automated extracting and summarizing information
    - Automatically analyze extracted information