CS 547: Foundation of Computer Security

S. Tripathy
IIT Patna

Previous Class

- Computer Security Strategy
- Challenges in policy to Implementation
  - Requirement Bugs, Design Bugs, Implementation Bugs

This Class

- Program security
  - Flaws: faults, and failures
  - Types of security flaws
  - Unintentional security flaws
    - Buffer overflows

Program Security

Program security is the first step to apply security in computing

Program Security Issues

- Issues:
  - How to keep program free from flaws?
  - How to protect computing resources from pgms with flaws?

- Issues of trust?
  - How trustworthy is that program you buy?
  - How to use it in its most secure way?
    - Partial answers:
      - Third-party evaluations
      - Liability and s/w warranties

Flaws, faults, and failures

- A flaw is a problem with a program
  - A security flaw is a problem that affects security in some way
    - Confidentiality, integrity, availability
- Two types of flaw:
  - faults and failures
    - A fault is a mistake "behind the scenes"
      - A fault is a *potential problem*
      - An error in the code, data, specification, process, etc.
    - A failure is when something actually goes wrong
      - You log in to the library's web site, and it shows you someone else's account

Types of security flaws

- One way to divide up security flaws is by genesis (where they came from)
- Some flaws are intentional
  - Malicious flaws are intentionally inserted to attack systems, either in general, or certain systems in particular
    - If it's meant to attack some particular system, we call it a targeted malicious flaw
  - Nonmalicious (but Intentional) flaws are often features that are meant to be in the system, and are correctly implemented, but nonetheless can cause a failure when used by an attacker

Types of Program Flaws

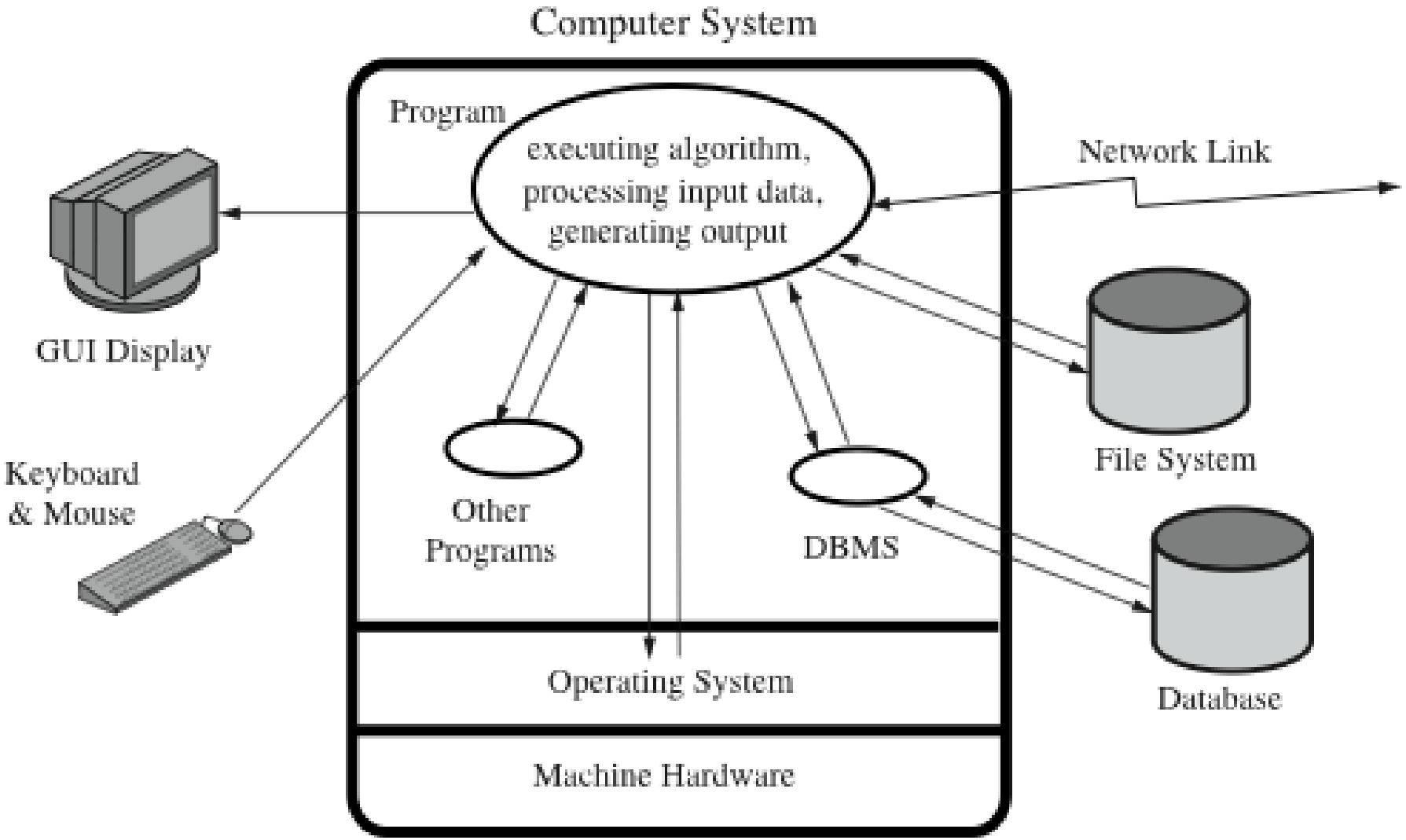
Taxonomy of pgm flaws:

- Intentional
  - Malicious
  - Nonmalicious
- Inadvertent
  - Validation error (incomplete or inconsistent)
    - e.g., incomplete or inconsistent input data
  - Domain error
    - e.g., using a variable value outside of its domain
  - Serialization and aliasing
    - serialization – e.g., in DBMSs or OSs
    - aliasing - one variable or some reference, when changed, has an indirect (usually unexpected) effect on some other data
  - Inadequate ID and authentication
  - Boundary condition violation, etc.

Unintentional program errors

- Most security flaws are caused by unintentional program errors
- will look at some of the most common sources of unintentional security flaws
  - Buffer overflows
  - Incomplete mediation
  - TOCTTOU errors (race conditions)

Abstract Program Model



Program Security Issues

- Many vulnerabilities result from poor programming practices
  - consequence from insufficient checking and validation of data and error codes
  - awareness of these issues is a critical step in writing more secure program code
- Program error categories:
  - insecure interaction between components
  - risky resource management
  - porous defences

Software Quality and Reliability

- Concerned with accidental failure of program
  - as a result of some theoretically random, unanticipated input, system interaction, or use of incorrect code
- SW quality can be improved using
  - structured design and testing to identify and eliminate the bugs from a program
  - Concern is how often they are triggered
  - failures are expected to follow some form of probability distribution

Software Security

- Attacker exploits the bugs that result in failure
- triggered by inputs that could differ dramatically from what is usually expected
- unlikely to be identified by common testing approaches

Ex.: memcpy(void *dst, const void *src, size_t n)

Defensive Programming

- programmers often make assumptions about the type of inputs a program will receive and the environment it executes in
  - assumptions need to be validated by the program and all potential failures handled gracefully and safely
  - programmers have to understand how failures can occur and the steps needed to reduce the chance of them occurring in their programs

2021 CWE Top 10

Rank	ID	Name
[1]	CWE-787	Out-of-bounds Write
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[3]	CWE-125	Out-of-bounds Read
[4]	CWE-20	Improper Input Validation
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[7]	CWE-416	Use After Free
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[9]	CWE-352	Cross-Site Request Forgery (CSRF)
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type

Handling Program Input

- input is any source of data from outside and whose value is not explicitly known by the programmer when the code was written
  - must identify all data sources
- incorrect handling is a very common failing
  - explicitly validate assumptions on size and type of values before use

Input Size & Buffer Overflow

- programmers often make assumptions about the maximum expected size of input
  - allocated buffer size is not confirmed
    - resulting in buffer overflow
- testing may not identify vulnerability
  - test inputs are unlikely to include large enough inputs to trigger the overflow
- safe coding treats all input as dangerous

Interpretation of Program Input

- program input may be binary or text
  - binary interpretation depends on encoding and is usually application specific
- there is an increasing variety of character sets being used
  - care is needed to identify just which set is being used and what characters are being read
- failure to validate may result in an exploitable vulnerability
  - Heartbleed OpenSSL bug is a recent example of a failure to check the validity of a binary input value

Buffer Overflow/Buffer Overrun

- A buffer overflow, also known as a buffer overrun, is defined in the NIST:

"*A condition at an interface under which more input can be placed into a buffer or data holding area than the capacity allocated, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system.*"

- The single most commonly exploited type of security flaw

Simple example:

- `#define LINELEN 120`
- `char buffer[LINELEN];`
- `gets(buffer);` ***or***
- `strcpy(buffer, argv[1]);`

Buffer Overflow

- Buffer overflow flaw —
  - often inadvertent (=>nonmalicious) but with serious security consequences
- Many languages require buffer size declaration
  - C language statement: char sample[10];
    - Execute statement: sample[i] = 'A'; where i=10
    - Out of bounds (0-9) → buffer overflow occurs
  - Some compilers don't check for exceeding bounds
    - C does *not* perform array bounds checking.
    - Similar problem caused by pointers
      - No reasonable way to define limits for pointers

Buffer Overflow

- Where does '*A*' go?
  - Depends on what is adjacent to 'sample[10]'
    - Affects user's data - overwrites user's data
    - Affects users code - changes user's instruction
    - Affects OS data - overwrites OS data
    - Affects OS code - changes OS instruction