Input Validation

(Review Questions)

- Why should we validate data input into a program?
  - i.e., What might happen if we do not validate input data?
    - See slides 2 through 5 for details
- How can data be input into a program?
  - i.e., What are the sources of external data?
    - See slide 6 for details
- What are the common ways to validate input data?
    - See slide 7 for details
- What should code do when invalid data is detected?
    - See slide 8 for details

Input Errors

- Example 1: $1 Billion typing error:
  - In 2005, a Japanese securities trader mistakenly sold 600,000 shares of stock at 1 yen each instead of selling one share for 600,000 yen.
- Example 2: $100,000 typing error:
  - A Norwegian woman mistyped her account number. Instead of typing her 11-digit account number, she accidentally typed an extra digit. The system discarded the extra digit, and transferred $100,000 to the (incorrect) account.
- Both errors above were preventable by simple input validation checks!
  - Example 1: Check the minimum price per share
  - Example 2: Check the account number has the correct number of digits

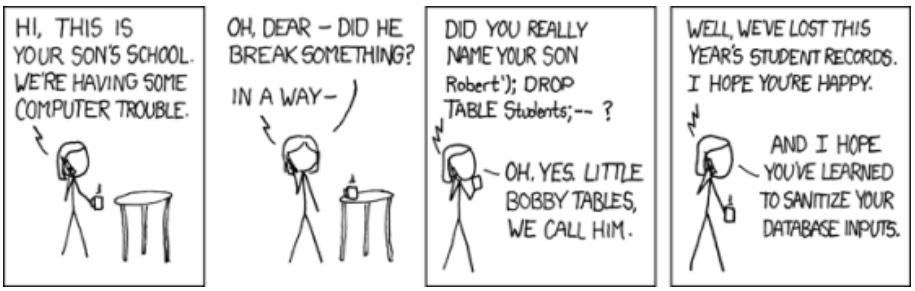
Input Errors cause Security Vulnerabilities

- Input errors can be caused by
  - accidental mistakes by trusted users
  - Malicious users looking to take advantage of flaws in the system
    - malicious user: one who intentionally crafts input data to cause programs to run unauthorized commands
    - Discuss: How can a malicious person take advantage of the input errors from the previous slide?

Malicious Input Error Attacks

- Credit cards stolen:
  - In Feb 2002, Jeremiah Jacks discovered at that Guess.com a properly-crafted URL allowed anyone to pull down 200,000+ names, credit card numbers and expiration dates in the site's customer database.
  - Known as a SQL-Injection attack
- Other common attacks that leverage poor input validation include:
  - Cross-site scripting – allows attackers to inject client side scripts into webpages to bypass access controls.
  - In-band signaling attacks- In older phone networks phone commands and voice data were sent over the same channel. It allowed for phone phreaking attacks where intentionally supplied commands were used to gain free calls or drop calls.

SQL Injection Attack



- Comic by XKCD

Main Points

- Programs often use external data
  - User input, file, database, network
- All external data that can enter your program can be a potential source of problems.
- Using external data without validation can make your system susceptible to security vulnerabilities.

Common ways to validate input data

1. **Range check (reasonableness check)** - numbers checked to ensure they are within a range of possible values, e.g.,
   - the value for month should lie between 1 and 12.
   - Stocks cannot be sold for less than 1 yen
2. **Length check:** ensure input is of appropriate length, e.g.,
   - US telephone number has 10 digits.
   - Bank account numbers are 11 digits long
3. **Type check:** input should be checked to ensure it is the data type expected, e.g.,
   - age must be integer.
4. **Format check** – Check that the data is in a specified format (template),
   - e.g., dates might be required to be in the format DD/MM/YYYY.
5. **Arithmetic Errors:** variables are checked for values that might cause problems such as
   - division by zero or integer overflow.

What to do if input has errors?

- When input errors are detected the program should immediately reject the request.
  - Do not attempt to interpret erroneous input into a correct one. Why?
    - Malicious user can craft input in a way so that the corrected version is an attack
- Use “deny-by-default” design principal
  - anything not explicitly permitted is forbidden.