Software Construction Laboratory

Week 5 Part 1

Lab 3

Mushi Zhou

Winter 2017

UCLA

Prof. Paul Eggert

Update on Assignment Due Dates

- Prof. Eggert has decided to extend due dates of all feature assignments for 6 days
- All assignments are due Friday nights
- Assignment 4 is not due until this Friday, 2/10
- You can refer to the assignment webpage for correct due dates

System Call Programming and debugging

Outline

- Buffers, Buffered I/O
- Why do we want to use buffer
- Buffer overruns, and techniques for avoiding them
- System Calls vs Library calls (Part2)
- How to use system calls in C (Part2)
- C and system programming (Part2)

What is Data Buffer

- In computer science, a data buffer is a region of a physical memory storage used to temporarily store data while it is being moved from one place to another
- Example:

- If you're watching a movie online, the web service will continually download the next 5 minutes or so into a buffer, that way your computer doesn't have to download the movie as you're watching it

- This is the same idea in programming

Why Do We Want to Use Data Buffer?

- The speed of reading and writing of the data may not be the same
- If there is no buffer, the faster one has to wait for the slower one for each iteration
- So if reading is faster, writing to the buffer can occur while reading is getting ready so that there are more data to read at the beginning
- If writing is faster, the writer does not have to wait there for reader to take the data, instead just puts data in the buffer and write more
- Buffer is also useful when transferring data between processes or programs
- This means using a buffer greatly increases data processing speed!

What Could Go Wrong with Data Buffer?

- What if writing is too fast that the entire buffer is filled?
- What if reading is too fast that it reads data that has not been written yet?

- Buffer overflow & underflow!

- Underflow is not too much of an concern, simply waiting can resolve the issue

Buffer Overflow

- A buffer overflow, or buffer overrun, is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations

- Why is this a great concern?
- What if the memory location adjacent stores executables, system files, important data, or stack or heap structures?

- Undefined behaviors, crushes, and also very dangerous
- Morris worm (One of the first Internet warms) is based on buffer overflow

Why Buffer Overflow Is Difficult to Prevent

- C/ C++ provide no built-in protection against accessing or overwriting data in any part of memory and do not automatically check that data written to an array is within the boundaries of that array

- C++ has libraries that allow programmer to manually check the issue, but C does not provide any library support for this

- Bounds checking can prevent buffer overflows, but requires additional code and processing time

Protective Countermeasures

- Choice of programming language
- Use of safe libraries
- Buffer overflow protection
- Pointer protection
- Executable space protection
- Address space layout randomization
- Deep packet inspection
- Testing

Choice of programming language

- Assembly and C/C++ are popular programming languages that are vulnerable to buffer overflow
- Languages that are strongly typed and don't allow direct memory access, such as Java, Python, and others, prevent buffer overflow from occurring in most cases
- Many programming languages other than C/C++ provide runtime checking and even compile-time checking which might send a warning or raise an exception instead of execute until something wrong happens

Address space layout randomization

- Address space layout randomization (ASLR)
- A computer security feature which involves arranging the positions of key data areas
- Usually includes arranging the base of the executable and position of libraries, heap, and stack, randomly in a process' address space.
- Can even further to the randomize virtual memory addresses
- Forces the attacker to tailor the exploitation attempt to the individual system, which make wide internet attack based on buffer overflow very difficult

Example of C Functions That Uses Data Buffer

- int getchar(void)
- int putchar(int char)

- These functions read from or write to a buffer that temporarily stores input/output
- This means when you ask user to input some characters from stdin and read them with getchar(), then all characters are stored in a buffer when the user presses enter, and then been read one by one by getchar()
- If you use a function without buffer to read, you will literately read one byte by one byte from stdin when the user presses enter