LAB 4

GitHub, General Feedback on PAs, Malloc, Hiding Sensitive Info in Apps

GITHUB

- Setup account
  - If you use your WSU account, you can get GitHub Education free
- Setup first repo for labs

GITHUB

- When to commit
  - Every time you create a function
    - i.e., you establish a function header and return a default value
    - Allows us to follow test-driven-development
  - Every time you write a function in its entirety
  - Update functions
  - Create a class
  - Anytime you make a change which could affect the rest of your program
- Commit Messages
  - Summary of the change made
  - Few words, but detailed
  - You will need to look at them later to know what went wrong
- Push once you are done with commits

INITIALIZING A STRUCT INLINE

Suppose we have:

```
typedef struct _test {
    int first;
    char * second;
    double third;
} Test;
```

We can initialize as such:

```
Test myStruct = {
    .first = 987,
    .second = "Something",
    .third = 123.456
};
```

COMPARISON WITH NULL

- NULL is used with a pointer, not a value typically
  - In C, NULL used as a value makes it appear as if var is a pointer
- NULL is not intended to be used as a value for numerical values
- nullptr will represent a null pointer in C++
- NULL provides ambiguity since it can equal 0 or a pointer, so use nullptr in C++
  - Note: NULL will work the same in most cases, but nullptr is more specific and prefered

ENUM (EX 1)

- We can use an enum to declare a value without having to remember which number means what and reduces the amount of required comments

```
typedef enum : int {
    kEXIT, // Exit
    kEDIT, // Edit List
    kDELETE, // Delete From List
    kLOAD, // Load from the file
    kSAVE, // Save to the file
} UserSelection;
```

ENUM (EX 2)

- We can use an enum to declare a value without having to remember which number means what and reduces the amount of required comments

```
typedef enum : int {
    OKAY, // Clean exit
    FAILED_TO_OPEN_FILE, // Failed to open a file
    FAILED_TO_CLOSE_FILE, // Failed to close a file
    FAILED_TO_ALLOCATE, // Failed to allocate memory
} ErrorCode;
```

RETURN ERROR CODES

- Suppose a function could fail because memory is not allocated or the infile was not open
- When we return 1 or 0 (true false), we do not know why it failed
  - Therefore, we can return an error code
  - As seen on the last page, we have a good code with multiple reasons for failure
  - Check the value returned at the caller

MIXING FEOF AND FGETS

- There is no need to mix feof and fgets
- If we use fgets, then we can simply use that as our condition in the while loop
- As feof could produce an inaccurate result since it asks where we are right now
  - Fgets will ensure that data exists and we do not risk crashing

MALLOC – WHEN WILL IT FAIL?

- On your system: probably never
- In the real-world:
  - When the device has limited memory
  - An attacker is maliciously sending a DOS attack
    - Fork-bomb (depletes memory and reduces availability of heap)
    - Many requests that use malloc
  - Using kernel space (kmalloc)

MALLOC FAILURE: EXAMPLE

- A new service has a single service since there are 100 users. The server has 100GB RAM. The system uses 30 GB on average.
- The service becomes popular and 10,000 users are now using the single server
- Suppose an average user consumes 7 MB after 1 week of usage
- @ 100 users, Each user gets about 700MB
- @ 10000 users, Each user gets about 7MB
  - Now, most users will be ok, but if there are many people who use more memory.
  - Now others users suffer and the same user does

MALLOC FAILURE: EXAMPLE

- An attacker sends a fork bomb instruction to the server to run as an admin
  - Now, hundreds of thousands of processes are running in a matter of seconds
  - Windows: %0|%0
  - Linux: :(){ :|:& };:
  - C: int main() { while(1) fork(); }
  - Pearl: perl -e "fork while fork" &
- This will take a lot of memory and therefore reduces the memory available for malloc

MALLOC FAILURE: POOR OS

- Poor OS
  - If the OS does not cleanup memory usage, then there could be 10 GB available, but they are in 5-40 byte segments. When you try to allocate 41+bytes in the heap it will fail. This is called fragmented memory. Modern main-stream OSes handle this to ensure it will not occur in practice
  - It can occur when there are a lot of calls to malloc and free, and it keep allocating at address 0xXXXXXXX, but it never goes back and see if we can move data around to defragment memory, so it is allocated as 0 1 2 3 4 2 3 5 instead of 0 1 2 3 4 5 6 7 where 2 and 3 could have been used, but it was unchecked

ARE LITERAL STRINGS READABLE IN BINARIES?

- Yes, every literal string you type, there will some human-readable form in the binary
- Common programs to extract this data
  - strings
  - hexdump
  - cat (but is not commonly used)

- Live demo of extracting a flag from a program.

LAB 4

LAB 4 - STACK

- A stack (programmatically) is like a singly linked list
  - Pointer to another node
  - Data
- There is a single pointer between the top node and the node below it

LAB 4 - TESTS

- We need tests for push, pop, isEmpty, top, peek
  - pushEmptyTest
  - pushNonEmptyTest
  - popEmptyTest
  - popNonEmptyTest
  - isEmptyEmptyTest
  - isEmptyNoEmptyTest
  - topEmptyTest
  - topNotEmptyTest

- Live Demo for writing peek test