LAB 4

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

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

FGETS

- Only use fgets, do not use fscanf, sscanf, or any of those horrible funcs
- Fgets will ensure we read the read the whole file
- Then, we use strtok to parse through the string
  - FIRST CALL: strtok(line, “,\n”)
  - OTHER CALLS: strtok(NULL, “,”)

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

REVIEW MAKENODE FROM CLASS

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

- Let's review the PDF

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 (if time)