

# Intro. to Secure Programming

## Study Groups at NCYU

Chih-Hsuan Yang(SCC)

[zxc25077667@pm.me](mailto:zxc25077667@pm.me)

April 21, 2021

# About me

- ▶ 楊志璿
- ▶ NSYSU Information security club founder
- ▶ Resume
- ▶ Linux, Modern C++

# Outline

Background

Programmer's qualities

- Binary search

- ReDoS

- Integer overflow

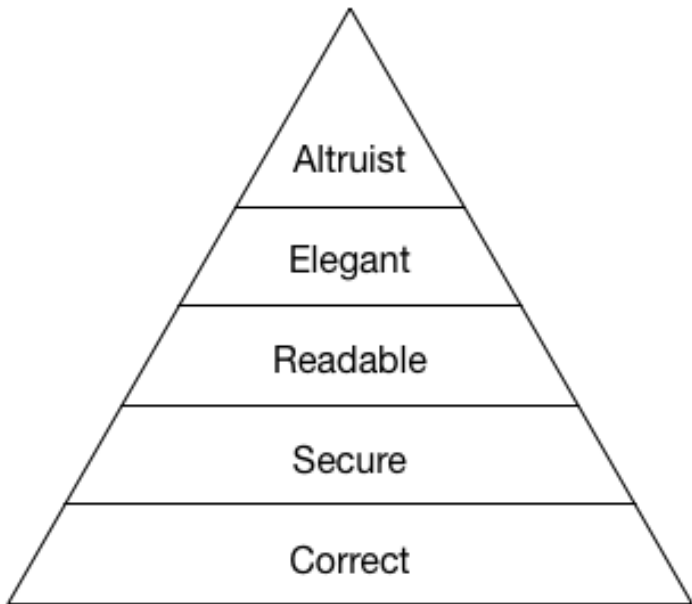
Memory safe

Real cyber attacks

Labs

# Background

## Maslow's pyramid of code review



# Maslow's pyramid of code review

- ▶ **Correct** : 做到預期的行為了嗎？能夠處理各式邊際狀況嗎？即便其他人修改程式碼後，主體的行為仍符合預期嗎？
- ▶ **Secure** : 面對各式輸入條件或攻擊，程式仍可正確運作嗎？
- ▶ **Readable** : 程式碼易於理解和維護嗎？
- ▶ **Elegant** : 程式碼夠「美」嗎？可以簡潔又清晰地解決問題嗎？
- ▶ **Altruist** : 除了滿足現有的狀況，軟體在日後能夠重用嗎？甚至能夠抽離一部分元件，給其他專案使用嗎？

## Programmer's qualities

# Binary search

```
1 int wrong(int *arr, size_t len, int target)
2 {
3     int begin = 0, end = len;
4     while (begin <= end)
5     {
6         int mid = (begin + end) / 2;
7         if (arr[mid] == target)
8             return mid;
9         else if (arr[mid] < target)
10             end = mid;
11         else
12             begin = mid;
13     }
14     return -1;
15 }
```



# Binary search

```
1 int correct(int *arr, size_t len, int target)
2 {
3     int begin = 0, end = len;
4     while (begin <= end)
5     {
6         int mid = (begin >> 1) + (end >> 1);
7         if (arr[mid] == target)
8             return mid;
9         else if (arr[mid] < target)
10             end = mid;
11         else
12             begin = mid;
13     }
14     return -1;
15 }
```

# Binary search

- ▶ 1946 idea
- ▶ 1960 mathematical analysis
- ▶ 1988 find bugs.

## Donald Knuth

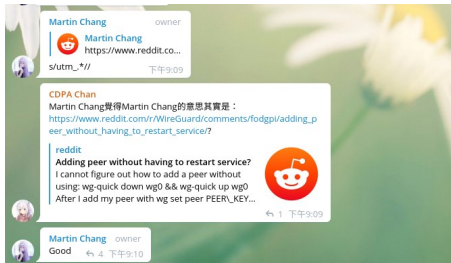
Although the basic idea of binary search is comparatively straightforward, the details can be surprisingly tricky.

## Appendix here - Donald Knuth

- ▶ T<sub>E</sub>X
- ▶ The Art of Computer Programming (TAOCP)

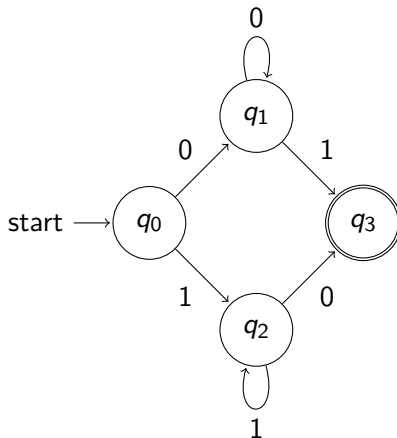


# ReDoS



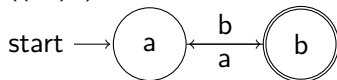
# Regex

- ▶ Regular expression
- ▶ Finite state machine



# Regex basic

## Halting problem

$$^+ ((ab)^*) + \$$$


# Input

abababababababababababa

## Halting problem

The engine will first try (ababababababababababab) but that fails because of that extra a. This causes catastrophic backtracking, because our pattern (ab)\*, in a show of good faith, will release one of it's captures (it will "backtrack") and let the outer pattern try again.

- ▶ (ababababababababababab) - Nope
- ▶ (ababababababababababab)(ab) - Nope
- ▶ (ababababababababababab)(abab) - Nope
- ▶ (ababababababababababab)(ab)(ab) - Nope
- ▶ (ababababababababababab)(ababab) - Nope
- ▶ (ababababababababababab)(abab)(ab) - Nope
- ▶ (ababababababababababab)(ab)(abab) - Nope
- ▶ (ababababababababababab)(ab)(ab)(ab) - Nope
- ▶ ...
- ▶ (ab)(ab)(ab)(ab)(ab)(ab)(ab)(ab)(ab)(ab)(ab)(ab) - Nope



# Halting problem

NP-Hard

$$dp[0] = 1$$

$$dp[N] = \sum_{i=0}^N dp[i] + dp[N - i]$$
$$\sim O(3^N)$$

# ReDoS

So, please check what you did.

User provides regex should have a timeout threshold.

Don't believe the user inputs.

# Appendix here

1. Halting problem
2. Dynamic programming
3. Finite state machine

# Halting problem

# Dynamic programming

# Finite state machine

# Integer overflow

Data Model					
Type	LP32	ILP32	LP64	ILP64	LLP64
char	8	8	8	8	8
short	16	16	16	16	16
int	16	32	32	64	32
long	32	32	64	64	32
long long	64	64	64	64	64
pointer	32	32	64	64	64

Memory safe



# Buffer overflow

# Memory leakage

## Real cyber attacks

# Labs