CTF.TamKy meeting #01

Luong Doan

April 30th, 2020

Overview of CTF

Overview of CTF i

1. What is CTF?
    1.1 Capture the Flag
    1.2 Information security competitions
2. Common types:
    2.1 **Jeopardy**
    2.2 Attack-Defence
    2.3 Mixed

Overview of CTF ii

Categories in Jeopardy CTF

1. Reverse Engineering
   - 1.1 Assembly/Machine Code
   - 1.2 Debugger (GDB)
2. Cryptography
   - 2.1 Hashing Functions
   - 2.2 RSA
3. Web Exploitation
   - 3.1 SQL Injection
   - 3.2 XSS
4. Binary Exploitation
   - 4.1 Buffer Overflow
   - 4.2 Return-Oriented Programming
5. Forensics
   - 5.1 Stegonagraphy

Overview of Computer Systems

Machine-Level Representation

1. Integer Representations
   - 1.1 Two's-complement
   - 1.2 Arithmetic Operators
   - 1.3 Relational Operators
   - 1.4 Logical Operators
   - 1.5 Bitwise Operators
   - 1.6 Assignment Operators
2. Floating Point
3. ASCII & Unicode
4. Machine-Level Code

Processor Architecture

1. x86 (Intel, AMD)
    1.1 Personal Computers & Laptops
2. ARM (Advanced RISC Machine)
    2.1 Smartphones & Tablets
    2.2 Embedded Systems & Supercomputers
3. MIPS (Microprocessor without Interlocked Pipelined Stages)
    3.1 Computer Architecture Courses in Universities

The Memory Hierarchy

1. Random-Access Memory (RAM)
   - 1.1 Static RAM (SRAM): Cache memory
   - 1.2 Dynamic RAM (DRAM): Main memory
2. Disk Storage
3. Cache memories
   - 3.1 CPU registers
   - 3.2 TLB
   - 3.3 L1 cache, L2 cache, L3 cache
   - 3.4 Virtual memory
   - 3.5 Disk cache
   - 3.6 Browser cache
   - 3.7 Web cache

Running Programs on a System i

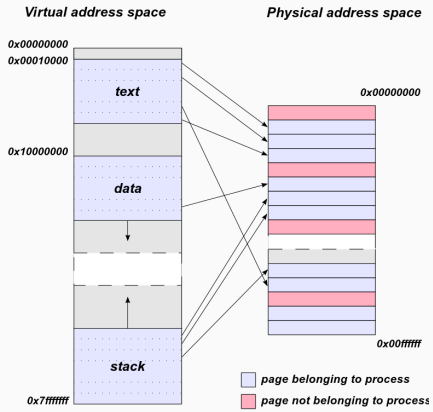
Exceptional Control Flow

1. Processes
2. Exceptions
3. Process Control
4. Signals
5. Non-Local Jumps

Running Programs on a System ii

Virtual Memory

1. Virtual address space



Hình 1: Virtual address space

2. Paging
3. Address Translation
4. Garbarge Collection

I/O & File System

1. I/O devices: Disks, Networks, Touch Screen, USB Flash Drive, Keyboard, Mouse, etc.
2. All I/O devices are modeled as files.
3. Low-Level Interface
   - 3.1 Opening Files
   - 3.2 Changing the Current Position
   - 3.3 Reading and Writing Files
   - 3.4 Closing Files
4. Three specific permissions in UNIX file system
   - 4.1 Read (r)
   - 4.2 Write (w)
   - 4.3 Execute (x)

Interaction and Communication Between Programs ii

Networking

1. Computer Networks & The Internet
2. The Client-Server Model

Interaction and Communication Between Programs iii

Concurrency & Parallelism

1. ”Concurrency is when two tasks overlap in execution.”
2. Processes, I/O Multiplexing & Threads
3. Concurrency vs Parallelism

C Programming

Basics i

Input:

```
int scanf ( const char * format, ... );
```

Output:

```
int printf ( const char * format, ... );
```

Compile and run:

```
$ gcc -o hello hello.c
$ ./hello
```

Basics ii

”Hello, World!”program:

```c
#include <stdio.h>

int main() {
    /* This is a comment */
    // This is also a comment
    printf("Hello, World! \n");
    return 0;
}
```

Data Types i

Integer Types

Type	Size	Value range
char	1 byte	0xffffff80 to 0x7f
short	2 byte	0xffff8000 to 0x7fff
int	4 byte	0x80000000 to 0x7fffffff
long	8 byte	0x8000000000000000 to 0x7fffffffffffffff

Data Types ii

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

int main(int argc, char** argv) {
    printf("CHAR_MAX    :   %d\n", CHAR_MAX);
    printf("CHAR_MIN    :   %d\n", CHAR_MIN);
    printf("SHORT_MAX   :   %d\n", SHRT_MAX);
    printf("SHORT_MIN   :   %d\n", SHRT_MIN);
    printf("INT_MAX     :   %d\n", INT_MAX);
    printf("INT_MIN     :   %d\n", INT_MIN);
    printf("LONG_MAX    :   %ld\n", LONG_MAX);
    printf("LONG_MIN    :   %ld\n", LONG_MIN);
    return 0;
}
```

Data Types iii

Floating-Point Types

Type	Size	Precision
float	4 byte	6 decimal places
double	8 byte	15 decimal places
long double	10 byte	18 decimal places

Data Types iv

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <float.h>

int main(int argc, char** argv) {
    printf("FLT_MAX              :   %g\n", FLT_MAX);
    printf("FLT precision value :   %d\n", FLT_DIG);
    printf("DBL_MAX              :   %g\n", DBL_MAX);
    printf("DBL precision value :   %d\n", DBL_DIG);
    printf("LDBL_MAX             :   %Lg\n", LDBL_MAX);
    printf("LDBL precision value:   %d\n", LDBL_DIG);
    return 0;
}
```

Reverse Engineering i

Use the GNU Debugger:

```
$ gdb -q hello
```

Dump of assembler code for function main:

```
0x000055555555463a <+0>:     push   %rbp
0x000055555555463b <+1>:     mov    %rsp,%rbp
0x000055555555463e <+4>:     lea    0x9f(%rip),%rdi
0x0000555555554645 <+11>:    callq  0x555555554510 <puts@plt>
0x000055555555464a <+16>:    mov    $0x0,%eax
0x000055555555464f <+21>:    pop    %rbp
0x0000555555554650 <+22>:    retq
```

References:

References: i

References:

1. Bryant, Randal E., O'Hallaron David Richard, and O'Hallaron David Richard. Computer systems: a programmer's perspective. Vol. 2. Upper Saddle River: Prentice Hall, 2003.
2. https://howtodoinjava.com/java/multi-threading/concurrency-vs-parallelism/
3. https://ctf101.org
4. https://ctftime.org
5. https://www.tutorialspoint.com/cprogramming/