

CS5231 System Security Midterm

INSTRUCTIONS

1. This test contains Three (3) questions and comprises Three (3) pages, including the this page.
2. ALL questions need to be answered. Answer all questions within the space in this booklet.
3. This is an OPEN BOOK examination.
(You are allowed to use ONE device in AIRPLANE mode to browse course related materials. No significant amount of typing or photo taking are allowed. These rules will be strictly enforced.)
4. The TOTAL marks for this exam is 25 marks.
5. Please write your Matric Number below (without your name).
6. Please be concise. Each short-answer question is expected to be answered by no more than 3 short sentences.

MATRICULATION NO: _____

1. Multi-Choice Questions (5 marks)

(a) (1 mark) Which of the following memory defenses rely on keep “secret” information in process memory? _____

- A. Stack canaries
- B. ASLR
- C. CFI with random tags
- D. Instruction Set Randomization

(b) ...

2. Fill-in Blanks (5 marks)

(a) (1 mark) Assume that you are debugging the following program using the GNU Debugger.

```
1  #include <stdio.h>
2  int f(int x, int y) {
3      return x+y;
4  }
5  int main() {
6      int a = 1, b = 2, c = 0;
7      c = f(a,b);
8      printf("%d + %d = %d\n", a, b, c);
9      return 0;
10 }
```

You have set a breakpoint at line 3 and have run to this breakpoint. The following table is the output for the command `disassemble main`.

```
(gdb) disassemble main
Dump of assembler code for function main:
0x000055555555161 <+0>:  endbr64
0x000055555555165 <+4>:  push    %rbp
0x000055555555166 <+5>:  mov     %rsp,%rbp
0x000055555555169 <+8>:  sub     $0x10,%rsp
0x00005555555516d <+12>: movl    $0x1,-0xc(%rbp)
0x000055555555174 <+19>: movl    $0x2,-0x8(%rbp)
0x00005555555517b <+26>: movl    $0x0,-0x4(%rbp)
0x000055555555182 <+33>: mov     -0x8(%rbp),%edx
0x000055555555185 <+36>: mov     -0xc(%rbp),%eax
0x000055555555188 <+39>: mov     %edx,%esi
0x00005555555518a <+41>: mov     %eax,%edi
0x00005555555518c <+43>: callq   0x55555555149 <f>
0x000055555555191 <+48>: mov     %eax,-0x4(%rbp)
0x000055555555194 <+51>: mov     -0x4(%rbp),%ecx
0x000055555555197 <+54>: mov     -0x8(%rbp),%edx
0x00005555555519a <+57>: mov     -0xc(%rbp),%eax
0x00005555555519d <+60>: mov     %eax,%esi
0x00005555555519f <+62>: lea     0xe5e(%rip),%rdi    # 0x555555556004
```

```

0x0000555555551a6 <+69>: mov    $0x0,%eax
0x0000555555551ab <+74>: callq 0x55555555050 <printf@plt>
0x0000555555551b0 <+79>: mov    $0x0,%eax
0x0000555555551b5 <+84>: leaveq
0x0000555555551b6 <+85>: retq
End of assembler dump.

```

What is the return address of function f? (Only fill in the last three digits) **0x000055555555191**

(b) ...

3. Short Answer Questions (15 marks)

(a) A utility program uses the following function, check format, to verify inputs from users. The utility program uses a global variable `buf` to receive inputs. It then calls `check format`, which copies the string in `buf` into a local character array variable `temp`.

```

int check_format()
{
    int i;
    char temp[8];

    i = 0;
    /* buf is a global variable holding
       user inputs */

    memcpy(temp, buf, strlen(buf))

    /* other operations */

    return 0;
}

```

Note: Here is the man page of `memcpy`

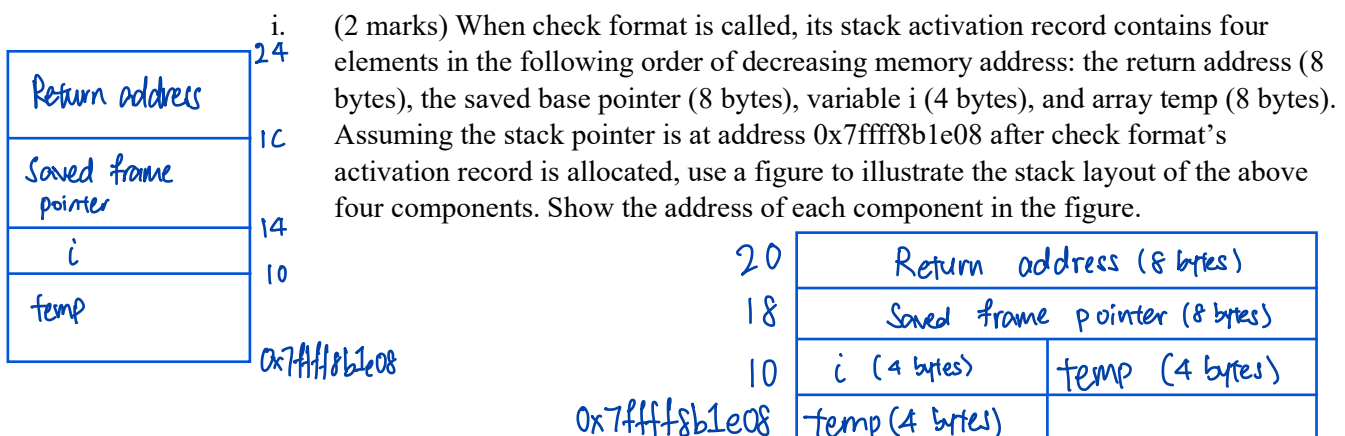
NAME: `memcpy` - copy memory area

SYNOPSIS

```
#include <string.h>
```

```
void *memcpy(void *dest, const void *src, size_t n);
```

DESCRIPTION: The `memcpy()` function copies `n` bytes from memory area `src` to memory area `dest`. The memory areas must not overlap.



- ii. (3 marks) Identify the memory error in the above program, and give an input in `buf` that changes the return address to a value you choose.

memory error: buffer overflow.

return address: 0xdeadbeefcafebabe

input: AAAAAAAAABBBBBBBBCCCC\xef\xbe\xad\xde\xbe\xba\xfe\xca

- iii. (3 marks) Discuss a defense technique and explain how it can prevent the attack.

(b)

Implement stack guard with a canary word.

Place a random canary word before the saved frame pointer and have it checked before returning. Stack smashing would require sequential write which would destroy the canary word, resulting in it being invalid and preventing the program to return to the user specified address.