ECE 568 – Computer Security

The Edward S. Rogers Sr. Department of Electrical and Computer Engineering

Mid-term Examination, Part 1, October 2019

Name	
Student #	

Answer all questions. Write your answers on the exam paper. Show your work.
Each question has a different assigned value, as indicated.

Permitted: one 8.5 x 11", two-sided page of notes.
No other printed or written material. No calculator.
NO PHOTOCOPIED MATERIAL
Total time: 50 minutes
Total marks available: 50 (roughly one mark per minute, with some extra time)
Verify that your exam has all the pages.
Only exams written in ink will be eligible for re-marking.

1 /25	2 /25	Total

Question 1: Buffer overflows [25 marks]

Program:

```
1:   int foo(char *arg)
2:   {
3:     char buf[16];
4:     int i, len;
5:
6:     len = strlen(arg);
7:
8:     if (len > 24)
9:       len = 24;
10:    for (i = 0; i <= len; i++)
11:      buf[i] = arg[i];
12:    return 0;
13:  }
14:
15:  int main(int argc, char *argv[])
16:  {
17:    char string[32];
18:
19:    strncpy(string, argv[1], 32);
20:    foo(argv[2]);
21:    return 0;
22:  }
```

Registers:

```
rbp            0x7fffffffe4a0   0x7fffffffe4a0
rsp            0x7fffffffe460   0x7fffffffe460
```

Stack:

```
0x7fffffffe460: 0x00000000      0x00000000      0xffffe83b      0x00007fff
0x7fffffffe470: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe480: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe490: 0x00000000      0x00000000      0x5327f500      0xae7022b9
0x7fffffffe4a0: 0xffffe4f0      0x00007fff      0x004006a6      0x00000000
0x7fffffffe4b0: 0xffffe5d8      0x00007fff      0x0040071d      0x00000003
0x7fffffffe4c0: 0x006f6f66      0x00000000      0x00000000      0x00000000
0x7fffffffe4d0: 0x00000000      0x00000000      0x00000000      0x00000000
```

Other info:

```
(gdb) p &buf
$1 = (char (*)[16]) 0x7fffffffe480
(gdb) p&i
$2 = (int *) 0x7fffffffe478
(gdb) p&len
$3 = (int *) 0x7fffffffe47c
(gdb) p &string
$4 = (char (*)[32]) 0x7fffffffe4c0
```

A program with a buffer overflow vulnerability is given above. The program is executed with an input passed in at the command line from the attacker. The state of the registers and stack when the program reaches line 6 is given. Answer the following questions (next page):

a) Are either the buffers `buf` or `string` vulnerable to a memory corruption attack? Please state your assumptions. [6 marks]

b) At what addresses on the stack are the return address of `main` and `foo` located? Explain your answer [10 marks]

c) Can an attacker exploit any vulnerability in this program to execute arbitrary code of the attacker's choice? Explain your answer [5 marks]

d) The attacker wants to get shellcode into the program but the attacker's shellcode is exactly 42 bytes long and can't fit into either buffer. Describe how the attacker can modify their shellcode to successfully solve this limitation. Use array notation to describe chunks of existing shellcode (i.e. shellcode[0-9] is the first 10 bytes of the old shellcode), and use pseudo-assembler to describe any new instruction you would insert into the shellcode [4 marks]:

Question 2: Fixing vulnerabilities [14 marks]

In this question you will be referring to the program in Question 1. For each proposed code change below, indicate with **Yes** or **No** whether the change fixes a vulnerability and/or introduces (i.e. adds) a new vulnerability. For clarity, the code changes have been bolded. For each answer, include a brief explanation [4 marks each]

i. Change line 19 to `strncpy(string, argv[1], 31);`

Fixes?	Adds?

ii. Change line 8 to `if (len >= 24)`

Fixes?	Adds?

iii. Change line 19 to **`snprintf(string, 32, argv[1],);`**

Fixes?	Adds?

iv. Change line 3 to char buf[**24**];

Fixes?	Adds?

e) Please explain if and how the following counter measures address the types of vulnerabilities in the program on page 2. For each vulnerability, please state if it **Completely** prevents the vulnerabilities from being exploited, **Mitigates** the vulnerability by making it harder to exploit or does **Nothing** to the vulnerability [3 marks each]:

i. Stackguard/Stack Canaries:

ii. Non-executable pages/DEP:

iii. Control-flow integrity (CFI):

ECE 568 – Computer Security

The Edward S. Rogers Sr. Department of Electrical and Computer Engineering

Mid-term Examination, Part 2, October 2019

Name	
Student #	

Answer all questions. Write your answers on the exam paper. Show your work.
Each question has a different assigned value, as indicated.

Permitted: one 8.5 x 11", two-sided page of notes.
No other printed or written material. No calculator.
NO PHOTOCOPIED MATERIAL
Total time: 50 minutes
Total marks available: 50 (roughly one mark per minute, with some extra time)
Verify that your exam has all the pages.
Only exams written in ink will be eligible for re-marking.

3 /25	4 /25	Total

Question 3: Cryptography [25 marks]

In a padding oracle attack, explain what role (if any) each of the following play in the attack. Justify your answer [4 marks each]

a) Cipher-block chaining (CBC):

b) Advanced Encryption Standard (AES):

c) Padding check:

d) Initial Vector (IV):

e) Suppose we have a protocol that is vulnerable to a padding oracle attack. We alter the padding as follows:

- Instead of using the same value for the pad, we use a counter starting with 1 up to *n* where *n* is the number of bytes of pad. For example, if there are 5 bytes of pad, the last 5 bytes of the last block will be 1, 2, 3, 4, 5.

Everything else remains the same. Suppose the last 6 bytes of the last 2 blocks of a message are as follows:

Block C_n

0x39	0xa5	0x14	0x68	0xa1	0x85

Block C_{n-1}

0x46	0x90	0xa8	0xb4	0x37	0x16

Answer the following [3 marks each]

i. By changing the last byte of Block C_{n-1} to 0xaa results in no padding error. What can the attacker infer is the plain text value of the last byte of the plain text of Block C_n? Explain your answer

ii. What should the attacker set the last byte of Block C_{n-1} to if she wants to decrypt the 2nd last byte of Block C_n ? Explain your answer

iii. In the worst case, how many decryptions must the attacker ask the server to do in order to recover the entire last block if the block size is 128 bytes? How many times more or less effort is this than it would take to bruteforce the key if the key is 128 bits? Explain your answer

Question 4: Miscellaneous

You observe an attacker sending the following string to a program you wrote.

```
"\xa8\xe4\xff\xbfAAAA\xaa\xe4\xff\xbf%04x%04x%04x%04x%n%244u%n\%08x\
n"
```

You suspect that the attacker is exploiting a format string vulnerability to overwrite a pointer in your program. Your computer is running 32-bit code.

a) At what address does the attacker think the pointer is located? Give the address in hex and provide an explanation [4 marks]

b) What value is the attacker overwriting the pointer with? Give the value in hex and provide an explanation [4 marks]

c) An attacker installs a key-logger on a victim's computer and is able to capture the victim's password. What aspect of the victim's security has been compromised? Circle the appropriate answer. [2 marks]:

i) Confidentiality ii) Integrity iii) Availability

d) Which attacks to Non-Executable pages prevent? Circle the appropriate answer: [3 marks/-1 per wrong answer]:

Return-into-libc	True	False
Code injection	True	False
Argument Overwrite	True	False

e) When a vendor provides a receipt for a purchase to the buyer, this is to guarantee what for the purchase? Circle the appropriate answer. [2 marks]:
i) Authentication ii) Integrity iii) Non-repudiation

f) What type of ciphers have the greatest encryption throughput? Circle the appropriate answer. [2 marks]:
i) Public-key Ciphers ii) Block Ciphers iii) Stream Ciphers

g) Compute the following values using modular arithmetic in a finite field as defined by the indicated modulus [2 marks each]:
i) $4 + 10 \bmod 11$

ii) $9 * 5 \bmod 13$

iii) $7 / 3 \bmod 13$

iv) $\log_5 4 \bmod 7$