King's College London

This paper is part of an examination of the College counting towards the award of a degree. Examinations are governed by the College Regulations under the authority of the Academic Board.

Degree Programmes	MSc, MSci
Module Code	7CCSMSEN
Module Title	Security Engineering
Examination Period	January 2019 (Period 1)
Time Allowed	Two hours
Rubric	ANSWER THREE OF FOUR QUESTIONS.
	All questions carry equal marks. If more than three questions are answered, the three answers with highest marks will count.
Calculators	Calculators are not permitted
Notes	Books, notes or other written material may not be brought into this examination

PLEASE DO NOT REMOVE THIS PAPER FROM THE EXAMINATION ROOM

1. Consider the following C code fragment:

```
 1 int
 2 main(int argc, char **argv)
 3 {
 4     (void) foo(argv[1]);
 5     exit(0);
 6 }
 7
 8 int
 9 foo(char *arg) { return bar(arg); }
10
11 int
12 bar(char *arg)
13 {
14
15     char lbuf[1024];
16
17     if (strlen(arg) >= 1024)
18         return -1;
19
20     memset(lbuf, 0, sizeof(lbuf));
21     puts("Welcome: ");
22     sprintf(lbuf, arg);
23     write(STDOUT_FILENO, lbuf, strlen(lbuf));
24     fflush(stdout);
25
26     return 0;
27 }
```

QUESTION 1 CONTINUES ON NEXT PAGE

SEE NEXT PAGE

a. Give a thorough description of the program's vulnerability. In particular, name the vulnerability (1 mark) and provide a detailed overview of its exploitation (3 marks). Then, identify and explain thoroughly all the components that are involved in the exploit (4 marks).

[8 marks]

b. How would an attacker exploit the vulnerability? Hint: describe in detail what the injection vector would look like (and what retaddr and retloc the attacker may use). Use symbolic values and addresses when needed (no need to write down the shellcode).

[10 marks]

c. Would StackGuard or a bounds checker mitigate the vulnerability (1 mark)? Explain clearly the reasons (4 marks).

[5 marks]

d. How can the program be fixed?

[2 marks]

2. Access Control

a. Access Control is an important concept in Computer Security.

i. What is a *subject* in access control terminology? Briefly justify your answer or provide an example.

[1 marks]

ii. What is the decision to grant or deny an access to a process based on? Briefly justify your answer or provide an example.

[1 marks]

iii. What is an Access Control List? Briefly justify your answer or provide an example.

[1 marks]

iv. What is associated a capability (i.e., a list of objects) to? Briefly justify your answer or provide an example.

[1 marks]

v. Describe the main properties of discretionary and mandatory access control policies.

[4 marks]

b. Suppose that in a system that enforces an information flow policy for integrity, subjects $s1$ and $s2$ have *high* security clearance (inherited from the associated principal), while objects $o1$, $o2$, $o3$, and $o4$ all have *medium* security sensitivity. Subjects $s3$ and $s4$ have *low* security clearance.

i. Using a diagram similar to the one shown in Fig. 1, indicate the allowed direction of information flow for this example, and show all the possible operations by all subjects/objects of the example by indicating whether each operation is allowed or denied by the example policy.

[9 marks]

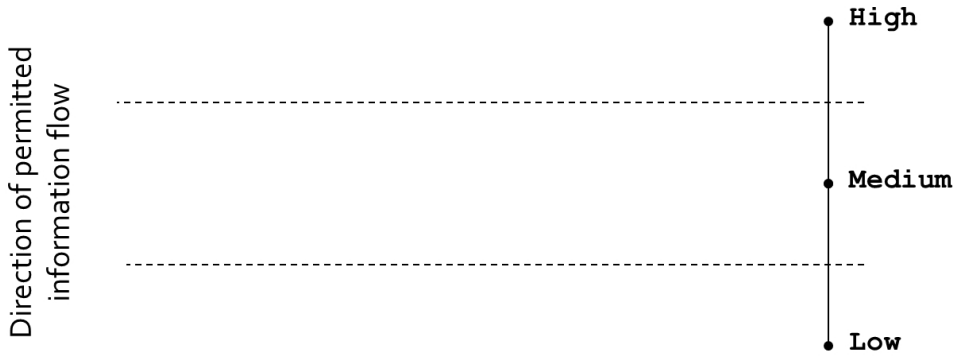

Figure 1: Information Flow Policy

ii. How are the *read* and *write* operations generically defined in terms of information flow?

[2 marks]

c. Read the following Unix command description.

> In Unix-like operating systems, `chmod` is the command and system call which may change the access permissions to file system objects (files and directories). It may also alter special mode flags.

i. Briefly describe the classes of operations/permissions and users in Unix-like OSes used for file access control.

[3 marks]

ii. Describe an example of a special mode flag in Unix-like OSes

[3 marks]

3. Consider the following C code fragment, which makes use of the `strcpy` version that truncates a copy after n bytes (i.e., `strncpy`):

```
1  int
2  main(int argc, char **argv)
3  {
4
5      char foobar[1024];
6
7      if (argc > 1)
8          strncpy(foobar, argv[1], strlen(argv[1]));
9
10     exit(1);
11 }
```

a. Does the program suffer from a memory corruption vulnerability? If not, explain the reasons. If yes, is it possible to *successfully exploit* this vulnerability? In other words, is it possible to provide specific input to such a program to take advantage of its vulnerability and thus execute arbitrary code (for instance, spawning a shell), on x86-32 architectures? If yes, explain how you would exploit it (high-level steps). If not, explain why and what you would change in the code to make it exploitable.

[5 marks]

QUESTION 3 CONTINUES ON NEXT PAGE

b. Assuming that the above code is vulnerable (or can be modified to become vulnerable) and that the vulnerability can be successfully exploited (or can be modified to be exploited), then consider the following x86 assembly code fragment, which may be used to exploit the previous vulnerability:

```
1  int
2  main(void)
3  {
4
5      __asm__(
6          "jmp ahead\n"
7          "back:\n"
8          "    popl %ebx\n"
9          "    movl %ebx, 0x8(%ebx)\n"
10         "    movl $0x0, %eax\n"
11         "    movb %al, 0x7(%ebx)\n"
12         "    movl %eax, 0xc(%ebx)\n"
13         "    movl %eax, %edx\n"
14         "    movl $0xb, %eax\n"
15         "    leal 0x8(%ebx), %ecx\n"
16         "    int $0x80\n"
17         "ahead:\n"
18         "    call back\n"
19         "    .string \"/bin/sh\""
20     );
21 }
```

i. Assuming the above assembly snippet (shown in Question 3 (b)) will be placed on the stack, what does the assembly code do? Add comments to each line and draw the stack layout **before** and **after** the considered instruction is executed. Note: you should clearly point out the direction the stack is growing towards.

[12 marks]

ii. An attacker creates a suitable injection vector to exploit the aforementioned memory error. To this end, he places the shellcode in the injection vector, pads it with his initials so as to create a message long enough to overflow `lbuf`; then the attacker adds the appropriate address at the right place and terminates the message with a `NULL` ('\0'). In other words, the injection vector looks as follows:

```
+----------------+--------    ---------+-----------+----+
|    shellcode    | lclclc ... lclclclc | 0xbff00b4r | \0 |
+----------------+--------    ---------+-----------+----+
```

Next, he runs the program giving this injection vector to it as its first argument. To his surprise, the attack fails. He asks you for help. State why the attack cannot work.

[2 marks]

iii. Show how the shellcode can be modified to make the attack possible. (Note: don't worry if you do not know the exact syntax of instructions; marks will be awarded for a clear explanation.)

[2 marks]

QUESTION 3 CONTINUES ON NEXT PAGE

c. Assume the aforementioned code is vulnerable (or can be modified to be so) and the vulnerability can be successfully exploited. State and describe what technique(s) would an attacker use to exploit the vulnerability shown at the beginning of the question, if the kernel enforces a non-executable stack protection (again, assume the small program shown at the beginning of the question is exploitable or can be modified to be exploited successfully)?

[4 marks]

4. Cross Site Scripting and SQL Injection attacks.

a. Cross Site Scripting is often abbreviated as XSS.

i. Briefly describe how Cross Site Scripting (XSS) works.

[4 marks]

ii. State what information an attacker can steal using XSS and why is it useful.

[3 marks]

iii. How can the effects of XSS be mitigated? Please outline limitations as well, if any.

[3 marks]

b. SQL Injection is a popular way of attacking applications that use SQL databases.

i. Briefly describe how SQL Injection works.

[4 marks]

ii. Apart from username and password input fields, which variables are candidates for SQL Injection?

[3 marks]

iii. What techniques can an application programmer use to mitigate the effects of SQL injection attacks? Please outline limitations as well, if any.

[4 marks]

QUESTION 4 CONTINUES ON NEXT PAGE

SEE NEXT PAGE

iv. An online shopping site `www.example.com/shop.php` takes a `name` from an URL parameter and constructs an SQL query as follows:

```
$query = "SELECT * FROM Users WHERE username = '$name'"
```

Construct an URL and the corresponding SQL query that delete all entries from the customers table[1]. The following character codes may be useful:

Encoding	ASCII value
%20	space
%27	'
%3b	;
%3d	=

[4 marks]

[1] In SQL, `DELETE` works just like `SELECT`.