NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 2 EXAMINATION 2022-2023

CE4067/CZ4067 – SOFTWARE SECURITY

Apr/May 2023 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 5 questions and comprises 7 pages.
2. Answer **ALL** questions.
3. This is a closed-book examination.
4. All questions carry equal marks.

---

1. Answer the following questions about the mitigation techniques studied in class.

   (a) Briefly describe the "W⊕X" protection and explain why it would prevent against stack smashing. Also suggest one way to bypass this protection.
   (5 marks)

   (b) Some web servers place a secret and unpredictable token value into an invisible field of HTML forms that are used to commit security-critical transactions. What security risk can such a token mitigate and why?
   (5 marks)

   (c) How is memory safety different from type safety? Provide a code example which is memory safe but not type safe.
   (5 marks)

   (d) What is flow-sensitive taint analysis? Provide a code example where flow-sensitivity makes a difference for the analysis result, i.e., flow-sensitive and flow-insensitive taint analyses produce different results.
   (5 marks)

2. The C program shown in Figure Q2a is vulnerable.

```
1  int main(void)
2  {
3    char *ptr_h;
4    char h[64];
5
6    ptr_h = getenv("HOME");
7
8    // Note: strlen("You home directory is: !") == 25
9    if(ptr_h != NULL && 64 - strlen(ptr_h) > 25) {
10     sprintf(h, "Your home directory is: %s!", ptr_h);
11     printf("%s\n", h);
12   }
13
14   return 0;
15 }
```

Figure Q2a: A C program printing the home directory.

(a) Name the vulnerability(ies) in this program and explain how an attacker may take advantage of the vulnerability(ies) to launch an attack.

(6 marks)

(b) You are only allowed to modify a single line at a time. Suggest TWO ways to fix the program.

(i) For each fix, state the corresponding line number and write down your modifications.

(4 marks)

(ii) Do the two fixes result in an equivalent program behavior? Explain your answer.

(4 marks)

(c) List at least THREE general defenses for the vulnerability(ies) and briefly explain why they are able to mitigate the issue(s).

(6 marks)

3. (a) The `snprintf()` function allows developers to limit the amount of data written to a buffer:

```
int snprintf(char * buffer, size_t size,
             const char * format, ...);
```

It composes a string with the same text that would be printed if `format` was used on `printf`, but instead of being printed, the content is stored as a C string in the buffer pointed by `buffer`. If the resulting string would be longer than `size-1` characters, the remaining characters are discarded and not stored, but counted for the value returned by the function. Briefly explain whether specifying a proper "`size`" limit will prevent the following types of vulnerabilities.

(i) Arbitrary reads (information leakage)

(2 marks)

(ii) Arbitrary overwrites

(2 marks)

(iii) Buffer overflows

(2 marks)

(b) Consider a double free attack that exploits the `unlink` function given in Figure Q3a.

```
1  #define unlink(P, BK, FD) {
2    FD = P->fd;    /* forward chunk */
3    BK = P->bk;    /* backward chunk */
4    FD->bk = BK;   /* update forward chunk's bk pointer */
5    BK->fd = FD;   /* updated backward chunk's fd pointer */
6  }
```

Figure Q3a: The unlink procedure of the Doug Lea malloc implementation.

(i) On a 32-bit machine, suppose we would like to overwrite a function pointer at `0x5655578c` to make it point to our shellcode at `0x22222222`. What values would you place in a ghost chunk tag to exploit Line 4 in Figure Q3a? Be specific and include the addresses you would use.

(5 marks)

(ii) Consider Line 5 in Figure Q3a. How would you construct the

Note: Question No. 3 continues on Page 4

shellcode to avoid this line from breaking your exploit?

(5 marks)

(iii) How would you modify the `unlink` procedure to verify that the bin is not corrupted? Please provide specific code to be added or removed.

(4 marks)

4. A typical way to test concurrent programs is to introduce delays, achieved by calls to the `sleep` function. For example, "`sleep(s1)`" in Figure Q4a makes the calling thread "Thread 1" sleep for `s1` second(s). Comparing with such (non-zero) delays, the time taken by other code execution is negligible. Therefore, different schedules can be simulated by introducing delays at various program locations. Assume `s1`, `s2`, and `s3` take values from $\{0, 5\}$, $\{0, 2, 4\}$, and $\{1, 3\}$, respectively.

```
1       sleep(s1);
2       if(access("tmpfile", W_OK)==0) {
3         sleep(s2);
4         fd=open("tmpfile", O_WRONLY);
5         write(fd, ...);
6       }
```

Thread 1

```
1       sleep(s3);
2       unlink("tmpfile");
3       link("/etc/passwd", "tmpfile");
```

Thread 2

Figure Q4a: A concurrent C program with two threads.

(a) Use the pairwise testing method to generate a minimal set of test schedules for the two threads shown in Figure Q4a. List the generated test schedules and explain your answer.

(7 marks)

(b) Under which test schedule(s), allowed by the given sleep time values, a TOCTTOU bug may be triggered? What is the security implication of this bug? Suggest a way to fix it.

(7 marks)

(c) What is the minimum number of test schedules generated from Part (a)

Note: Question No. 4 continues on Page 5

needed to cover all possible interleavings of the two threads? Treat `s1`, `s2`, and `s3` as symbolic variables. Provide a condition for each interleaving as a symbolic formula, such that the interleaving is enabled if and only if the condition holds.

(6 marks)

5. (a) List ONE example of meta-characters in SQL. Describe and explain what possible security issue(s) may arise from the meta-character provided.

(5 marks)

(b) Cross-site scripting attacks are also called XSS attacks. Compare Reflected XSS with DOM-based XSS. Explain their key similarities and differences. Describe ONE possible defense against XSS.

(5 marks)

(c) Figure Q5a shows the code snippet of a vulnerable server program that handles requests for querying student grades. Figure Q5b shows a sample XML database that stores the user ID, password, and grade information of all students for each course. The server program accepts HTTP GET requests and looks up student information from the XML database according to the provided course ID and password.

(i) Name the types of vulnerability(ies) contained in the code shown in Figure Q5a. Specify the line number(s) of the vulnerable code.

(4 marks)

(ii) Suppose the server program is accessible from http://www.vulnerable.com/query_grade. Provide a security test case, in the form of a URL query string, that demonstrates how an attacker may query arbitrary grades stored in the XML database in Figure Q5b without knowing the lecturer's password.

(3 marks)

(iii) Suggest at least TWO defense techniques which can prevent the vulnerability(ies) in the code shown in Figure Q5a.

(3 marks)

Note: Question No. 5 continues on Page 6

```
1  protected void doGet(HttpServletRequest request,
2                       HttpServletResponse response) {
3    // Access XML document
4    File inputFile = newFile("grades.xml");
5    DocumentBuilderFactory factory =
       ↪ DocumentBuilderFactory.newInstance();
6    DocumentBuilder builder = factory.newDocumentBuilder();
7    Document doc = builder.parse(inputFile);
8    XPath xPath = XPathFactory.newInstance().newXPath();
9
10   // Get inputs from HTTP GET Parameters
11   String cid = request.getParameter("cid");
12   String pwd = request.getParameter("pwd");
13
14   // Construct & execute a query to grades.xml
15   String query = "/grades/course[@cid='" + cid + "' and
       ↪ @lecturer_pwd=" + pwd + "]/student";
16   NodeList nodeList = (NodeList)
       ↪ xPath.compile(query).evaluate(doc,
       ↪ XPathConstants.NODESET);
17
18   // Send nodeList as the response to the client
19   ...
20 }
```

Figure Q5a: The Java servlet program for querying student grades.

Note: Question No. 5 continues on Page 7

```
1  <?xmlversion="1.0" encoding="UTF-8"?>
2  <grades>
3    <course cid='CS4061' lecturer_pwd='812813'>
4      <student>
5        <uid>John11</uid>
6        <pwd>300300</pwd>
7        <grade>B</grade>
8      </student>
9      <student>
10       <uid>Jack12</uid>
11       <pwd>004004</pwd>
12       <grade>F</grade>
13     </student>
14     ...
15   </course>
16   <course cid='CS4062' lecturer_pwd='812813'>
17     <student>
18       <uid>Jack12</uid>
19       <pwd>004004</pwd>
20       <grade>F</grade>
21     </student>
22     ...
23   </course>
24   ...
25 </grades>
```

Figure Q5b: The grades.xml file storing the grade information of students.

END OF PAPER

CE4067 SOFTWARE SECURITY
CZ4067 SOFTWARE SECURITY

Please read the following instructions carefully:

1. **Please do not turn over the question paper until you are told to do so. Disciplinary action may be taken against you if you do so.**

2. You are not allowed to leave the examination hall unless accompanied by an invigilator. You may raise your hand if you need to communicate with the invigilator.

3. Please write your Matriculation Number on the front of the answer book.

4. Please indicate clearly in the answer book (at the appropriate place) if you are continuing the answer to a question elsewhere in the book.