UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR10058 COMPUTER SECURITY

Thursday 14th May 2015

09:30 to 11:30

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

CALCULATORS MAY NOT BE USED IN THIS EXAMINATION

Year 3 Courses

Convener: S. Viglas
External Examiners: A. Cohn, T. Field

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. **[Cryptography]** Let $(\mathcal{E}_{32}, \mathcal{D}_{32})$ be a secure symmetric cipher with 32-bits key size and 32-bits message size. That is

$$\begin{aligned} \mathcal{E}_{32}: &\quad \{0,1\}^{32} \times \{0,1\}^{32} \to \{0,1\}^{32} \\ \mathcal{D}_{32}: &\quad \{0,1\}^{32} \times \{0,1\}^{32} \to \{0,1\}^{32} \end{aligned}$$

We want to use this cipher to build a new symmetric cipher $(\mathcal{E}_{64}, \mathcal{D}_{64})$ that will encrypt 64-bits messages under 64-bits keys:

$$\begin{aligned} \mathcal{E}_{64}: &\quad \{0,1\}^{64} \times \{0,1\}^{64} \to \{0,1\}^{64} \\ \mathcal{D}_{64}: &\quad \{0,1\}^{64} \times \{0,1\}^{64} \to \{0,1\}^{64} \end{aligned}$$

We consider the following encryption algorithm. To encrypt a message M under a key K, we first split M into two 32-bits parts M_1 and M_2, and K into two 32-bits parts K_1 and K_2. The ciphertext C is then obtained by concatenating the encryption of M_1 under K_1 using $\mathcal{E}_{32}$, with the encryption of M_2 under K_2 using $\mathcal{E}_{32}$. Thus, formally the $\mathcal{E}_{64}$ encryption algorithm is defined as follows. For all $M_1, M_2, K_1, K_2 \in \{0,1\}^{32}$

$$\mathcal{E}_{64}(K_1||K_2, M_1||M_2) = \mathcal{E}_{32}(K_1, M_1)||\mathcal{E}_{32}(K_2, M_2)$$

(a) What is the corresponding decryption algorithm? Prove that the consistency property is satisfied. *[5 marks]*

(b) Consider the following game.

- In a first phase, the attacker chooses a few plaintext messages $P_1, \ldots, P_n$ and gets back the corresponding ciphertexts $C_1, \ldots, C_n$ under some key K that he does not know. The attacker gets to know that C_1 is the ciphertext corresponding to $P_1, \ldots, C_n$ is the ciphertext corresponding to P_n.
- In a second phase the attacker builds two different messages M and M' and gets back C which is the encryption under K either of M or M'. But now, the attacker doesn't know if the plaintext underlying C is M or M' and he has to guess it.

Informally, a symmetric cipher is said to be subject to a *chosen plaintext attack* if the attacker can win this game; that is if he can guess (with high probability) which of M or M' is the plaintext corresponding to the ciphertext C that he sees. Show that the new cipher $(\mathcal{E}_{64}, \mathcal{D}_{64})$ is subject to a chosen plaintext attack even though $(\mathcal{E}_{32}, \mathcal{D}_{32})$ is secure.

[**Hint:** from any two messages P_1, P_2, and corresponding ciphertexts C_1 and C_2 under some secret key K, the attacker can craft two new messages M and M' for which he knows the corresponding ciphertexts under K without knowing the key K] *[10 marks]*

QUESTION CONTINUES ON NEXT PAGE

(c) A symmetric cipher is said to be vulnerable to a *known plaintext attack* if given a plaintext message M and its corresponding ciphertext C under some key K not known to the attacker, the attacker can recover the key K in a reasonable amount of time (that is significantly less than by a brute-force attack).

i. Explain what is meant by a brute-force attack. What is its complexity? [*3 marks*]

ii. Give an algorithm that recovers the key K from M and C in significantly less time than a brute-force attack on the 64-bits key space. Give your algorithm's complexity. [*7 marks*]

2. [**Memory safety**] Consider the following code where `EOF` is the End-Of-File character:

```
int vuln() {
  char name[20];
  int c, i = 0;
  printf(``What's your name?\n'');
  while ((c = getchar()) != EOF) {
    name[i] = c;
    i = i+1;
  }
  name[i] = '\0';
  printf(``Nice to meet you, %s!\n'', name);
  return 0;
}
```

(a) Depict the stack frame of a call to `vuln()` before the execution of the `while` loop. You will assume that the code is executed on a 32-bit machine, with the size of the `int` data type being 4 bytes, and the size of the `char` data type being 1 byte. [*5 marks*]

(b) This code violates memory safety. What is the precise vulnerability in the code? How can it be exploited? [*5 marks*]

(c) The assembly code generated by the compiler for this function includes a function epilogue that looks like this:

```
movl %eax, 0x1eaff00d  // copies in %eax the value stored
                       // at an address in the .data
                       // section of memory

movl %eax, 0(%eax)

movl %esp, %ebp        // restore the stack pointer to
                       // the top of the frame

cmp %eax, -4(%esp)     // checks if %eax equals value at
                       // address %esp-4

jne <stack_smashed>    // if they are different, jump to
                       // code that kills the program

ret
```

QUESTION CONTINUES ON NEXT PAGE

Also, before `main()` runs, the compiler inserted code that generates a random 32-bit number which is stored at address `0x1eaff00d`; every function prologue pushes this onto the stack, and every function epilogue has the check shown above. What buffer overflow defence seen in class has the compiler implemented in this way? [*5 marks*]

(d) What kind of buffer overflows will this detect? What kind won't it detect? Justify your answer. [*5 marks*]

(e) Will the defence that the compiler inserted detect your attack from question (b)? Justify your answer. [*5 marks*]

3. [**Web security**] MyBook has implemented a new feature for finding friends. Users of MyBook can enter search string such as "robin" to find people to befriend. MyBook also tracks the most popular searches. When the user visits a URL of the form:

```
https://www.mybook.com/friendfinder.php?name=robin
```

MyBook runs the following (in pseudocode):

```
name = getParameterFromUrl(url, 'name');
increment_number_of_searches(name);
command = ``SELECT username FROM users WHERE name = ''' + name + ````;
results = execSQLForTable(command, ``users'');
html = ``<p>You searched for:  `` + name + ``.</p><p>'' + results + ``</p>'';
send_to_client(html);
```

This code runs as follows. It first extracts "robin" from the URL. It then issues an SQL query to the users table. The users table contains the username, password (in cleartext), and name of all of the MyBook users. Finally, the code sends back the results to the user.

When a MyBook user asks for the most popular searches, MyBook goes through the database containing the number of searches for each search string, and returns the top 100 search strings sorted by number of searches.

Consider the following MyBook users:

Alice - She knows Eve, so she will visit any link Eve sends to her. However, Alice will not ask to view the top 100 search queries.

Bob - He does not know Eve, so would not visit any link recommended by Eve. However, he would be curious of viewing the top 100 search queries every so often.

Charlie - He does not know Eve, so would not visit any link recommended by Eve. Also, he will never ask to view the top 100 search queries.

Eve - She is malicious.

(a) For each of the following goals, say if Eve could achieve that goal. If your answer is yes, then give the name of the attack technique she could use, and briefly describe your attack. If your answer is no, justify it. You will not assume any software vulnerability or design flaw others than those explicitly described in the statement of this exercise.

i. Eve wants to steal Alice's MyBook cookie. [*5 marks*]

ii. Eve wants to steal Bob's MyBook cookie. [*5 marks*]

QUESTION CONTINUES ON NEXT PAGE

iii. Eve wants to steal Charlie's MyBook cookie. [*5 marks*]

iv. Eve wants to steal Charlie's password. [*5 marks*]

(b) A security expert recommends to MyBook that they add an unpredictable CSRF token to the search form and URL. Thus the search URL now looks like:

```
https://www.mybook.com/friendfinder.php?token=1A0B743FC08DA37A&name=robin
```

The code is now modified to first check that the token is correct; if it is not correct then the the rest of the code is not executed and the page doesn't load. Nothing is changed in the code for the top 100 search strings. Which of goals i-iv have now become impossible? Justify your answer. [*5 marks*]