CE4062/CZ4062 Computer Security

Tutorial 3 – Software Security (2)

1. Answer the following questions
   a. What is the root cause of format string vulnerability? What are the possible consequences?

   b. How to prevent integer overflow vulnerabilities?

   c. What is the scripting vulnerability?

2. Consider the fragment of a C program in Figure Q2. The program has a vulnerability that would allow an attacker to cause the program to disclose the content of the variable "`secret`" at runtime. We assume that the attacker has no access to the exact implementation of the '`get_secret()`' function so the attack has to work regardless of how the function '`get_secret()`' is implemented.
   a. Explain how the attack mentioned above works. You do not need to produce the exact input to the program that would trigger the attack. It is sufficient to explain the strategy of the attack. Explain why the attack works.

   b. The vulnerability above can be fixed by modifying just one statement in the program without changing its functionality. Show which statement you should modify and how you would modify it to fix the vulnerability. Show the C code of the proposed solution

```
int main(int argc, char* argv[]) {
  int uid1 = x12345;
  int secret = get_secret();
  int uid2 = x56789;
  char str[256];
  if (argc < 2)
    return 1;
  strncpy(str, argv[1], 255);
  str[255] = '\0';
  printf("Welcome");
  printf(str);

  return 0;
}
```

Figure Q2

3. You are developing a web service, which accepts the email title '`title`' and body '`body`' from users, and forwards them to `fake-addr@ntu.edu.sg`. This is achieved by the following program in Figure Q3. Identify the security problems in this piece of program

4. The following program in Figure Q4 implements a function in a network socket: '`get_two_vars`'. It receives two packets, and concatenates the data into a buffer. Use an example to show this program has integer overflow vulnerability. Note the first integer in the received buffer from '`recv`' denotes the size of the buffer.

```
void send_mail(char* body, char* title) {
  FILE* mail_stdin;
  char buf[512];
  sprintf(buf, "mail -s \“Subject: %s\" fake-addr@ntu.edu.sg", title);
  mail_stdin = popen(buf, "w");
  fprintf(mail_stdin, body);
  pclose(mail_stdin);
}
```

Figure Q3

```
int get_two_vars(int sock, char *out, int len){
    char buf1[512], buf2[512];
    int size1, size2;
    int size;

    if(recv(sock, buf1, sizeof(buf1), 0) < 0)
        return -1;
    if(recv(sock, buf2, sizeof(buf2), 0) < 0)
        return -1;

    memcpy(&size1, buf1, sizeof(int));
    memcpy(&size2, buf2, sizeof(int));
    size = size1 + size2;
    if(size > len)
        return -1;
    memcpy(out, buf1, size1);
    memcpy(out + size1, buf2, size2);
    return size;
}
```

Figure Q4