LiTH, Linköpings tekniska högskola
IDA, Institutionen för datavetenskap
Ulf Kargén

Written exam

TDDC90 Software Security

2026-01-13

Permissible aids

Dictionary (printed, NOT electronic)

Teacher on duty

Ulf Kargén, 013-285876

Instructions and grading

There are 6 questions on the exam. Your grade will depend on the total points you score. The maximum number of points is 32. The following grading scale is preliminary and might be adjusted during grading. You may answer in Swedish or English.

Grade	3	4	5
Points required	16	23	27

Question 1: Secure software development (4 points)

a) Explain how the *Privilege Separation* pattern works. When is it applicable? (2 points)

b) Explain the concept of *attack surface reduction* and give a practical example of it. (2 points)

Question 2: Software reviews (3 points)

Name each of the roles in a *Software inspection* and briefly explain their responsibilities.

Question 3: Memory safety (6 points)

a) For each of the three exploit techniques below, state whether DEP is effective in mitigating the technique. Motivate each answer in a sentence or two. Answers without motivation gives no points. (3 points)

   i. ROP

   ii. NOP-sleds

   iii. return-to-libc

b) For each of the three exploit techniques below, state whether CFI is effective in mitigating the technique. Motivate each answer in a sentence or two. Answers without motivation gives no points. (3 points)

   i. ROP

   ii. Information disclosure via an out-of-bounds read

   iii. return-to-libc

Question 4: Web security (6 points)

a) Use pseudo-code and English/Swedish to explain what an *insecure deserialization* vulnerability is, and how it can be exploited by attackers. (3 points)

b) Name and explain a web attack that can lead to arbitrary code execution on the *client* (i.e., in a user's web browser). (2 points)

c) Briefly explain what a *supply chain vulnerability* is. (1 point)

Question 5: Security testing (6 points)

a) What is the benefit of using a *sanitizer* such as AddressSanitizer during fuzzing? What is the downside? (2 points)

b) Generation-based fuzzers typically do not need seed inputs. Explain what purpose the seed inputs fulfil in mutation-based fuzzing, and why this is not needed in a generation-based fuzzer. (2 points)

c) Greybox fuzzers, such as AFL, are sometimes also called *evolutionary fuzzers*. Explain why. (2 points)

Question 6: Static analysis (7 points)

Assume `int` denotes integers with an absolute value that can be arbitrarily large (so no integer overflows need to be considered). Consider the following procedure (where `=` stands for assignment and `==` stands for equality).

```
1  int foo(int x, int y){
2      int c = 0;
3      if(y > 0){
4          c = 1;
5      }else{
6          c = -1;
7      }
8      int rslt = 0;
9      int i = x;
10     while(i > 0){
11       if(c > 0){
12         rslt = rslt + i;
13       }else{
14         rslt = rslt - i;
15       }
16       i = i - 1;
17     }
18     assert(i <= 0);
19     assert((x < 0) || (2 * rslt * c == x * (x+1)));
20     return rslt;
21   }
```

Questions:

1. Symbolic execution:
   - How many feasible paths does the method `foo` have if the argument `x` can assume any of the 11 values in `[-5,5]` and `y` any of the 5 values in `[-2,2]`? (1 point)
   - Give in SSA form, without discussing its satisfiability, the path condition taking the `else` branch of the `if` at line 3, making exactly one iteration of the `while` loop with the `else` branch of the `if` at line 11, satisfying the first assertion and violating the second one. Observe there is one path condition following this description. It might not be satisfiable, but it should follow the program control flow. (2 points)
2. Abstract interpretation: Annotate, after convergence of a most precise value analysis, the start of each line in `foo` with an abstract element associated to each one of the defined variables (including the unknown values passed to `foo`). For each line, and for each variable or argument, give the interval domain obtained after convergence. (2 points)
3. Give, without justification, the weakest condition P such that executing the sequence of three assignments `z= x - 2*y; x= z + x; y= 2*(z - x);` (where `x`, `y` and `z` are integer variables with no integer overflows) from a configuration satisfying P will always result in values satisfying the condition ($Q : x == y$ and $z \leq x$). (2 points).