LiTH, Linköpings tekniska högskola
IDA, Institutionen för datavetenskap
Nahid Shahmehri

Written exam

TDDC90 Software Security

2007-12-21

Permissible aids

Dictionary (printed, NOT electronic)

Teacher on duty

David Byers, 013-282821, 0708-282821

Instructions

The exam is divided into two parts.

There are eight questions in part one. You should answer all of them.

There are four questions in part two. You should answer only one of them. If you fail to follow this instruction and answer more than one question in part two, we will randomly choose one of your answers to grade and disregard the others.

You can get a maximum of 32 points in part one and 6 points in part two, giving a maximum total of 38 points on the exam.

You may answer in Swedish or English.

Grading

Your grade will depend on the total number of points you accumulate on the exam.

The following grading scale is preliminary. It might be adjusted during grading.

Grade	3	4	5
Points required	20	27	33

Part one

Question 1: Security requirements (2 points)

Explain what SMART+ requirements are and how they can be used.

Question 2: Fuzz testing (2 points)

Which is better: fuzzing or static analysis? Elaborate and motivate your answer.

Question 3: Privilege separation (2 points)

Explain what privilege separation is, and how it helps improve security in software.

Question 4: Threat modeling (2 points)

Explain what threat modeling is and how it can be used in software development.

Question 5: Common Criteria (6 points)

What are security assurance requirements in Common Criteria and how are they used? If you use any further terminology from the Common Criteria, briefly explain each term.

Question 6: Static analysis (6 points)

Assume that we have the following abstract values:

- `(?)` denoting the set of all integers {…,-2,-1,0,1,2,…}
- `(+)` denoting the set of all positive integers {1,2,3,…}
- `(0)` denoting the set {0}
- `(-)` denoting the set of all negative integers {-1,-2,-3,…}

Assume further that we have the following piece of code

```
if ( x > 0) then
    x = 2 * x
else
    x = x * x
fi
```

What can be said about the value of **`x`** after execution of the program if we make a sound static analysis assuming initially that:

(a) **`x == (?)`**

(b) **`x == (+)`**

Continued on next page

(c) `x == (0)`

(d) `x == (-)`

Several sound values are possible in general. The best possible value is required for maximum number of points.

Question 7: Preventing exploits (6 points)

ProPolice (newer versions are known as SSP) includes the following three mechanisms that protect against buffer overflows: a canary is placed between the saved frame pointer and return address on the stack; local variables are arranged so that buffers are stored at higher addresses than pointers (i.e. pointers are higher on the stack than buffers); and function arguments are copied to the current stack frame from where they were placed by the caller.

(a) For each of the three mechanisms listed above explain how the mechanism protects against buffer overflows. Your answer must include a characterization of the buffer overflows each mechanism targets.

(b) Explain one attack that ProPolice does not protect against.

(c) Describe a protection mechanism that would prevent successful exploitation using the attack you describe in (b).

Question 8: RMF (6 points)

Explain the five stages of activities in a risk analysis framework (RMF). At what time in the software lifecycle is RMF used? Elaborate your answer with an example.

Part two

Important

Only answer one question in this section. If you answer more than one then we will choose one of your answers at random to grade and disregard the others.

Question 9: Security requirements (6 points)

Are security requirements functional or non-function requirements? Motivate your answer and give examples.

Question 10: Fuzz testing (6 points)

Explain the difference between white-box and black-box fuzz testing. Give a simple example demonstrating the two approaches. Discuss the merits and drawbacks of each approach.

Question 11: PaX (6 points)

PaX is a security add-on to Linux that incorporates a number of useful techniques. It implements something similar to W⊕X protection (which, among other things, results in a non-executable stack) and address space layout randomization (ASLR).

(a) On a platform with 32 bit addressing, the effectiveness of PaX is limited. Why? How can an attacker exploit this? Do the same limitations apply on platforms with 64 bit addressing? Motivate your answer.

(b) If there is an format string vulnerability that allows the attacker to examine the stack of the protected program, then PaX is quite easy to bypass. Explain how.

Question 12: Security processes and best practices (6 points)

The Sustainable Software Security Process (S^3P), CLASP and the Secure Development Lifecycle (SDL) are examples of approaches to secure software development. Choose two of these approaches and explain how they work and how they are related to security best practices.