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

Distance exam

TDDC90 Software Security

2021-03-18

Teacher on duty

Ulf Kargén, ulf.kargen@liu.se, 013-285876

Instructions and grading

There are 7 questions on the exam. Your grade will depend on the total points you score. The maximum number of points is 36. 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	19	27	32

Answers should be submitted through Lisam as a single PDF or ASCII text (.txt) document before the end of the exam time. If you don't have access to a good software for making technical drawings that you are already familiar with, it is recommended to draw figures by hand and scan/photograph them. Figures could either be integrated into the PDF, or submitted as separate files in JPEG or PNG format. If you chose the latter approach, file names should be of the format Figure1, Figure2, and so on. All attached figures should be clearly referred to in the text by their file name. To facilitate easier grading, it is preferred that you express formulas, program code, etc. as text, and not as handwritten figures.

You are allowed to use any aid, but **all kinds of collaboration with others is strictly forbidden**. Also, **copying any part of an answer from another source will be considered plagiarism**. The questions will be checked for plagiarism and sharing of answers between students using Urkund, and any suspected cheating will be reported to the university disciplinary board. **By taking the exam, you solemnly promise to abide by the above rules.**

In the event that you experience technical difficulties with Lisam that prevent you from submitting, it is allowable to submit answers via email. However, this should only be used as a last resort if Lisam for whatever reason would stop functioning.

The distance exam will not be anonymous due to the exceptional COVID-19 situation.

Ulf Kargén will be available to answer questions during the exam via email and phone.

Question 1: Secure software development (4 points)

Consider the CORAS method for risk analysis. If you would want to integrate the other two risk analysis methods discussed in the course – *misuse cases* and *attack trees* – into CORAS, in which step of CORAS would each respective method fit best? For each of *misuse cases* and *attack trees*, explain in which step of CORAS it would fit best, and clearly motivate how it would contribute to the CORAS workflow in that step. Only answer with one CORAS step per method.

Question 2: Exploits and mitigations (5 points)

Consider the two techniques *NOP-sleds* and *register trampolines*.

a) What is the purpose of using one of these techniques in an exploit?

b) Compare the two techniques in terms of their pros and cons.

c) Consider a scenario where you would have to place your shellcode on the heap instead of the stack. *In the general case*, would one of the above two techniques be significantly superior to the other in this scenario? If so, which one? Motivate your answer.

Question 3: Design patterns (2 points)

Consider a secret government storage facility with a number of individual storage vaults for sensitive materials of various kinds. For example, there is one vault with spare parts for secret stealth helicopters, one vault with handgun munitions, and one vault with frozen alien remains from downed UFOs. Knowledge of the contents of each vault is on a strict need-to-know basis. For example, only engineers involved in the stealth helicopter project is allowed to know about the helicopter spare parts, only facility security staff is to know about the type and quantities of stockpiled munitions, and only scientists involved in UFO research is allowed to know about the frozen aliens. (However, it is possible to have access to multiple vaults. One could, for example, be a part of both the stealth helicopter and UFO research teams.)

Imagine that you have been tasked with creating a software for compiling an inventory report, specifying the types and quantities of items in each vault. The software is to take a user's credentials, and based on these create a report detailing only the contents of the vaults that this specific user is allowed to know about.

Which of the design patterns *secure factory* and *secure chain of responsibility* would be most suitable to base the software design around? Clearly motivate your answer.

Question 4: Web security (6 points)

a) Briefly explain why using *salts* is important when storing passwords for a web app.

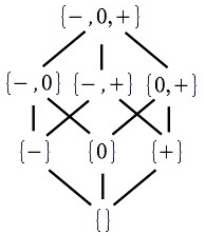
b) Which of *SQL injection* and *command injection* would you argue is the more severe type of vulnerability? Motivate your answer.

c) Explain how CSRF-tokens work. Using pseudocode and English/Swedish, also explain how CSRF-tokens can be implemented in a web app. You can use pseudocode resembling PHP, or any other server-side language you are familiar with, as long as it is clear how the (pseudo)code works.

Question 5: Static analysis (7 points)

Assume `int` denotes integers with an absolute value that can be arbitrarily large (i.e., no integer overflows). Consider the following two procedures.

```
1  void foo(int x){
2    if(x < 0){
3      x= -x;
4    }
5    int y= 0;
6    int z= x;
7    while(x > 0){
8      x--;
9      y++;
10     assert(0 < y);
11   }
12   assert(y == z);
13 }
```



We aim to check the assertions `(0 < y)` and `(y == z)` respectively at lines 10 and 12 in procedure `foo`. Questions:

1. Symbolic execution:
   (a) Give the SSA formula for the path condition that corresponds to the path starting at line 1 of procedure `foo` and performing one iteration of the loop before violating the assertion at line 12. (1 pt)
   (b) Given value of `x`, how many paths perform loop iterations and violate the assertion at line 12? Is any of them satisfiable? Explain. (2 pt)
2. Abstract interpretation: Annotate, after convergence of the analysis, each line in `foo` with the abstract element associated to the defined variables. (1pt)
3. The following part has 3 questions. The three answers result in a minimum of 0 pt and a maximum of 3 pts. Each wrong answer counts negative. E.g., one correct answer (1 * 1 pt), one wrong (1 * -1 pt) and not answering one (1 * 0 pt) results in a score of 0 pt out of the 3 possible points. Giving two wrong answers (2 * -1 pt) and one correct (1 * 1 pt) gives 0 points. State, without justification, whether each of the following Hoare-triples is true or false.
   (a) $\{y = 0\}$ `while(x > 0){x--; y++;}` $\{y = 0\}$
   (b) $\{y + x > 0\}$ `while(x > 0){x--; y++;}` $\{y > 0\}$
   (c) $\{x = 2 \text{ and } y = 2\}$ `while(x > 0){x--; y++;}` $\{x = y\}$

Question 6: Security testing (6 points)

a) Briefly explain two general reasons (i.e. not related to specific vulnerability types) why automated fuzzing of web applications is often harder than fuzzing e.g. a desktop program written in C.

b) Give an example of a testing target (SUT) where blackbox mutational fuzzing would be superior to all other fuzzing techniques discussed in the course (blackbox generational, greybox, whitebox/concolic testing). Clearly motivate your answer and state any assumptions you might make. Make sure to explain why each of the other fuzzing techniques would be unsuitable.

c) A generation based fuzzer generally requires two components to work: A grammar and a set of fuzzing heuristics. Explain the purpose of both these components.

Question 7: Vulnerabilities in C/C++ programs (6 points)

The code on the next page shows a function `read_stream`, which is part of a software for reading a data stream from a remote sensor. The sensor system uses a simple one-way packet-based radio transmission protocol. Since the communication is one way, there is no guaranteed delivery of packets (like in, for example, TCP). Instead, `read_stream` attempts to assemble as much of a data stream from the sensor as possible, and simply emits warnings when packets are lost. The function reads the stream one packet at a time, using the API function `get_packet` (see documentation in the code), and returns the complete assembled stream as a malloc-allocated buffer. The size of the stream is outputted to the parameter `stream_size_out`.

The code contains a serious security bug. Identify the bug in the code, explain how an attacker who is capable of transmitting fake sensor streams could trigger it (you don't need to show a full exploit), and finally show how the code should be updated to fix the bug.

```c
#define MAX_PACKET_SIZE 1024
struct STREAM_T;

/* Gets the next packet in the stream 's'.
   Packet data is written into 'output' and the size of the data is
   returned. The maximum size of packet data is 1024.
   If the next packet in the stream was lost due to a transmission error,
   -1 is returned. A return value of 0 denotes end of stream. */
int get_packet(struct STREAM_T* s, char* output);
```

```c
#define BUF_SIZE 1048576

char* read_stream(struct STREAM_T* s, size_t* stream_size_out)
{
   // Buffer for assembling stream
   char buffer[BUF_SIZE];

   // Used for reading one packet
   char packet_data[MAX_PACKET_SIZE];

   // Points to the current insert-position in 'buffer'
   char* buffer_pos = buffer;

   char* final_stream = NULL;
   size_t total_size = 0;
   int result = 0;

   do {
      result = get_packet(s, buffer);
      if(result < 0) {
         printf("WARNING: Data lost at offset %lu\n", total_size);
      } else if(total_size + result > BUF_SIZE) {
         // Note that we don't need to worry about integer overflows here
         // since 'result' can be at most 1024 and BUF_SIZE is always
         // necessarily much smaller than SIZE_MAX.
         printf("ERROR: Too much data\n");
         return NULL;
      } else {
         memcpy(buffer_pos, packet_data, result);
         buffer_pos += result;
      }
      total_size += result;
   } while(result != 0);

   final_stream = malloc(total_size);
   memcpy(final_stream, buffer, total_size);

   // Write size to variable pointed to by 'stream_size_out'
   *stream_size_out = total_size;

   return final_stream;
}
```