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

Distance exam

TDDC90 Software Security

2021-01-16

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)

a) For each of the following CORAS artefacts, state in which of the CORAS steps that the artefact is produced:
   - asset list
   - asset ranking
   - risk evaluation matrix
   - consequence scale
   - likelihood scale

b) Come up with a small example scenario, and go through the CORAS steps from (a) to produce the corresponding artefacts. Your answer will be graded based on the completeness and correctness of produced artefacts. Your example should include at least *two* different assets.

Question 2: Exploits and mitigations (5 points)

a) Most implementations of stack cookies also reorder the position of local variables on the stack in a specific way. Explain in what way variables are reordered, and the motivation for doing this.

b) Give an example scenario where ASLR would make exploiting a *use-after-free* bug harder. Provide sufficient technical details in your example to clearly motivate why ASLR is effective in your scenario.

Question 3: Design patterns (2 points)

Imagine that you are tasked with designing a special piece of software for viewing classified electronic documents. The document format allows tagging individual parts of documents as Top Secret, Secret, Confidential, or Unclassified. The document viewer must be able to hide classified parts of documents, where the credentials of the user determine if he/she is allowed to view a part of a document classified at a specific level. Moreover, the decision to show or hide document parts based on credentials should be based on a security policy, which must be possible to change during runtime (using a special admin interface to the software).

Based on the above requirements, suggest a secure design pattern that is suitable for implementing the decision-making logic that determines whether a document part should be shown or not. Clearly motivate your answer.

Question 4: Web security (6 points)

It is common that organizations have both a publicly-facing web server, and one or more web servers that are only accessible from their internal network. Consider such a setup, where the publicly-facing web server is also connected to the internal network, but where the web app served by the public server is not designed to interface with internal web servers in any way.

For each of the following vulnerability types, explain whether or not exploiting such a vulnerability in the public web server could allow an attacker to view content on internal web servers. Use a few sentences per vulnerability type to motivate your answer.

i. Command injection

ii. SQL injection

iii. XXE

iv. Reflected/Stored XSS

v. Insecure deserialization

vi. CSRF

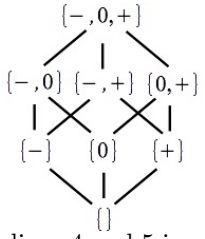
Note that here we only consider vulnerabilities that allow attackers to *directly* view content on internal web servers in their browsers. Multi-step attacks, where the attacker, for example, uses some exploit to plant a backdoor on the web server, or steals login credentials from employees in order to login to the internal network, are out of scope for this question.

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    int z= bar(x);
3
4    assert(0 < z);
5
6    assert(z < 20);
7  }
```

```
1  int bar(int y){
2    if(y < 0)
3      y= -y;
4    if(y > 10)
5      y= 10;
6    return y;
7  }
```



We aim to check the assertions `(0 < z)` and `(z < 20)` respectively at lines 4 and 5 in procedure `foo`. Questions:

1. Symbolic execution:
   (a) Give a path condition formula that corresponds to the path starting at procedure `foo`, calling procedure `bar` at line `foo.2`, following the `else` branch at the `if`-statement at line `bar.2` and the `then` branch at the `if`-statement at line `bar.4` and violating the assertion `0 < z` at line `foo.4`. As usual, treat procedure parameters (such as `y` in `bar`) as variables. (1 pt)
   (b) Is the path formula above satisfiable? is it enough to show that the assertion at line `foo.4` is never violated? Explain. (1 pt)
2. Abstract interpretation: Annotate, after convergence of the analysis, each line in `foo` and `bar` with the abstract element associated to each variable with a scope containing that line (i.e., `x` and `z` in `foo` and `y` in `bar`). (1pt)
3. The following part has 4 questions. The four answers result in a minimum of 0 pt and a maximum of 4 pts. Each wrong answer counts negative. E.g., two correct answers (2 * 1 pt), one wrong (1 * -1 pt) and not answering one (1 * 0 pt) results in a score of 1 pt out of the 4 possible points. Giving three wrong answers (3 * -1 pt) and one correct (1 * 1 pt) gives 0 points. We will use `y` to mean the input to procedure `bar` and `ret` to mean the integer (recall there are no integer overflows here) returned by `bar` at line `bar.6`. In an effort to establish the assertion at line `foo.6`, we study the following 4 Hoare-triples. State, and briefly explain, whether each of the following triples is true or false.
   (a) $\{y \geq -10\}$ `bar(y)` $\{ret > 0\}$
   (b) $\{y < -10\}$ `bar(y)` $\{ret \geq 10\}$
   (c) $\{true\}$ `bar(y)` $\{-15 \leq ret \leq 15\}$
   (d) $\{true\}$ `bar(y)` $\{1 \leq ret < 15\}$

Question 6: Security testing (6 points)

a) Recently, so-called hybrid fuzzing has been proposed as a way to combine greybox fuzzing and concolic testing. When the greybox fuzzer has repeatedly failed to "flip" a certain branch, the concolic system is instead invoked to try to solve the branch condition. If a solution is found, it is added to the seed pool of the fuzzer, and the fuzzing is resumed. This setup could, at least in theory, be more efficient than spending the same computational resources on either component (greybox fuzzer or concolic tester) alone. Explain why, and clearly motivate your reasoning.

b) Imagine that you are tasked with creating a black box fuzzer for a JavaScript interpreter. Which fuzzing strategy (mutation or generation) would be most appropriate? Clearly motivate your answer.

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

The code on the next page shows a function `process_array` that reads an array from a network stream. It assembles the array fragments received in each packet into a consecutive array, which is subsequently processed. (The nature of this processing is not relevant here.) The data format used stipulates that each element of the array should always be exactly 16 bytes. The function might be used to receive data from untrusted network streams.

The code contains a serious security bug. Identify the bug in the code, and explain how it should be fixed.

```c
#define ENTRY_SIZE 16
#define PACKET_LIST_SZ 1000

/*
Retrieves one packet from the network stream associated with 'handle'.
Returns packet data as a malloc-allocated buffer, and writes the
size of the data to 'size_out'. (The size of the allocated buffer
always corresponds exactly with the size written to 'size_out'.)
Returns NULL if the stream has been closed. */
char* get_packet(int handle, size_t* size_out);

struct PacketInfo {
   size_t size;
   char* data;
};

/*
Retrieves and processes an array from the network stream associated with
'handle'. Returns 0 on success and 1 on error. */
int process_array(int handle) {
   size_t total_entries = 0;
   char* final_array = NULL;

   char* packet = NULL;
   size_t packet_size;

   struct PacketInfo packet_list[PACKET_LIST_SZ];
   size_t list_index = 0;

   /* Retrieve all packets and accumulate them into 'packet_list' */
   while( (packet = get_packet(handle, &packet_size)) != NULL ) {
      if(list_index == PACKET_LIST_SZ) {
         printf("ERROR: Packet list full!\n");
         return 1;
      }
      packet_list[list_index].data = packet;
      packet_list[list_index].size = packet_size;
      list_index++;

      total_entries += (packet_size / ENTRY_SIZE);
   }

   /* Assemble array from packets */

   /* Note that an integer overflow is impossible here, as that would imply
   that the size of all packet data is greater than the entire virtual
   memory! */
   final_array = malloc(total_entries * ENTRY_SIZE);

   char* current_data_end = final_array;

   for(size_t i = 0; i < list_index; i++) {
      memcpy(current_data_end, packet_list[i].data, packet_list[i].size);
      current_data_end += packet_list[i].size;
      free(packet_list[i].data);
   }

   /* Do something with 'final_array' ... */

   free(final_array);
   return 0;
}
```