Examination Software Security 2018/19

Module/course code:	Software Security
Date:	23 January 2019
Time:	13:45-15:45
Instructors:	Erik Tews

Type of test:

- Closed book

Allowed material and aids during the test:

- None

Additional remarks:

- Read these instructions and the questions carefully! If the questions are unclear, you can ask for clarification. Always write down how you understood this clarification with your answer.
- Make sure you answered all parts of a question and not just some of them.
- Please make sure that your name and student number appear on all answer sheet
- Try to give precise answers using appropriate terminology and always give a reason for your answer.
- Unreadable or extremely long answers will not be marked.
- Give your answers in English.
- The exam consists of 6 pages including this cover sheet.
- There are 12 questions in total, all of them with the same weight for the final mark.

1 C bugs

Consider the following C function:

```
int * readAll() {
        int total, i;
        int * values;
        printf("How many values?\n");
        readInt(&total);
        if (total < 0) {
                printf("Illegal input\n");
                return NULL;
        }
        values = malloc(total * sizeof(int));
        if (values == NULL) {
                printf("malloc failed!\n");
                return NULL;
        }
        for (i = 0; i < total; i++) {
                readInt(&values[i]);
        }
        return values;
}
```

readInt(int *) is a function that reads an integer and it will never fail.

What is the bug in readAll?

2 Cppcheck

Assume that you develop a program for extracting information such as the resolution and other data from JPEG images in C with a few other developers. You would like to use Cppcheck to spot bugs in your source code during development. What would be the advantage of using the tool as part of a build and test pipeline on a server versus running the tool locally from time to time on the developers machine?

3 Constant time code

Consider the following fragment of C-code:

```
int calculate(int a) {
        int r = a / 256;
        if (r == 0) {
                return 1;
        } else {
                return 64;
        }
}
```

Does it run in constant time? When you think that depends on the compiler, platt form and other settings, assume that it compiled to the following code on x64:

```
0000000000001125 <calculate>:
    1125:       55                      push   %rbp
    1126:       48 89 e5                mov    %rsp,%rbp
    1129:       89 7d ec                mov    %edi,-0x14(%rbp)
    112c:       8b 45 ec                mov    -0x14(%rbp),%eax
    112f:       8d 90 ff 00 00 00       lea    0xff(%rax),%edx
    1135:       85 c0                   test   %eax,%eax
    1137:       0f 48 c2                cmovs  %edx,%eax
    113a:       c1 f8 08                sar    $0x8,%eax
    113d:       89 45 fc                mov    %eax,-0x4(%rbp)
    1140:       83 7d fc 00             cmpl   $0x0,-0x4(%rbp)
    1144:       75 07                   jne    114d <calculate+0x28>
    1146:       b8 01 00 00 00          mov    $0x1,%eax
    114b:       eb 05                   jmp    1152 <calculate+0x2d>
    114d:       b8 40 00 00 00          mov    $0x40,%eax
    1152:       5d                      pop    %rbp
    1153:       c3                      retq
```

4 Missing check for the return value of malloc

Consider the following program:

```
#include <stdlib.h>

int main(int argc, char** argv) {
        int i;
        int * t;
        t = malloc(100 * sizeof(int));
        for (i = 0; i < 100; i++) {
                t[i] = i;
        }
        free(t);
        return 0;
}
```

When you run it, it works just fine and exits without any error. However there is a bug, malloc might fail, returning NULL. When you would run the program under valgrind, will it find that bug?

5 AFL

Will afl be able to find the bug in the previous program?

6 CBMC

Consider the following program:

```
#include <stdio.h>
#include <assert.h>

int main(int argc, char** argv) {
        int sum = 0;;
        int total, i, t;
        printf("How many numbers?\n");
        scanf("%d", &total);
        if (total < 0) {
                printf("invalid input\n");
                return 1;
        }
        for (int i = 0; i < total; i++) {
                sum += 1;
        }
        printf("The sum is %d\n", sum);
        assert(sum == total);
        return 0;
}
```

Will CBMC be able to verify (quickly) that the assert statement is always true, or is it likely that CBMC will fail?

7 AFL with sint

In one of your assignments, you used AFL to fuzz your sint implementation. Compared to just using random inputs for the program, what does AFL try to optimize?

8 Rust and memory management

What is the advantage of the "borrow-checker" memory management of Rust compared to other languages such as Java or Python that use a garbage collector?

9 Rust coding

Consider the following lines of Rust code:

```
let mut v = vec![1,2,3];
v.push(25);
let v2 = v;
println!("{}", v[2]);
```

When you try compile an run it,

- Will it compile and run just fine?
- Or will it fail to compiled?
- Or will it compile just fine but then fail when you try to run it?

And in case of problems, also state which line causes the problem and why.

10 Rust and null-pointers

Rust does not have a null concept. What language construction is typically used to emulate it?

11 OWASP ZAP

Assume that you managed to build a web application. After a user has created an account, he can login, upload pictures, share them with other users and also comment on those pictures. You are developing this application in a DevOps environment. What kind of problem will you encounter when you run a standard ZAP (baseline) scan against the application without any additional configuration?

12 Design for security policies

One of the bullet points on the slides with guidelines for more secure software was “design for security policies”. Explain what that means and how the concept could for example be applied for the web application described in the previous question.