2020-01-22 - M-CS - Software Security - 201600051

Course: M-CS-201600051-1B M-CS Software Security 201600051

Number of questions: 13

Generated on: Jan 31, 2020

Contents: Pages:

- A. Voorpagina **1**
- B. Vragen **6**

2020-01-22 - M-CS - Software Security - 201600051

Course: M-CS Software Security 201600051

The exam consists of 13 questions and you have 120 minutes to answer them. Some require text as answer, some of them ask you to submit source code. There are 100 points in total, so when you solve a question with X points in X minutes, then you are making good progress.

For technical questions of the software use, raise your mouse. For contents related questions raise your hand.

It is not allowed to have an electronic communication device in a useable mode (except the computer you are writing the exam on) at any time during the exam.

Please don't leave the room in the last 20 minutes before the exam ends so that you don't disturb other students who are still busy with the exam.

The URL that are permitted for this exam are:

https://utwente.remindotoets.nl/

https://www.utwente.nl/en/telt/learning/intranet/books/

https://play.rust-lang.org/

https://www.rust-lang.org/learn

Good Luck!

Number of questions: 13

1
8 pt.

Assume that you have a C program that has a bug so that an integer, that is used as an array index isn't properly check, so that an array in this program is sometimes accessed outside it's array boundaries. The result is then often a Segmentation Fault, causing the program to terminate abnormaly.

When you access an array outside it's boundaries in Rust, the result is always a panic. Why is the panic in Rust better from a security point of view than a Segmentation Fault you get very often in C programs?

2
8 pt.

Why is C not a type safe language? Give an example about something you can do in C that would not be possible in a type safe language.

3
6 pt.

One common source of security problems in C programs is memory corruption that occurs during string handling. One solution that was presented in the lecture is to replace strncpy/strncat in the code with strlcpy/strlcat. Why is strlcpy/strlcat considered to be safer than strncpy/strncat?

4
6 pt.

Usually, memory in C is allocated using malloc. An alternative function to allocate memory is calloc. What kind of common problem or bug does using calloc instead of malloc prevent?

5
8 pt.

One of the tools you used in the lecture to find bugs in C programs. Name a bug that you can detect using Valgrind, that you would usually not spot when you would run the same program with the same input just from the command line. When possible, provide some C (pseudo-)code as well to illustrate the problem.

6
8 pt.

As long as you don't use "*unsafe*" in your Rust programs, your Rust programs are memory safe and no memory corruption will ever occur. Why is it still useful to fuzz your Rust programs with tools such as afl.rs or other fuzzers and even when your programs will never segfault, what kind of bugs may fuzzers find in your Rust programs?

7 Consider the following Rust program:

10 pt.

```
struct NamesList { l: Vec<String> }impl NamesList {
    fn new() -> NamesList {
        NamesList{ l: Vec::new() }
    }
    fn add(&mut self, s: String) {
        self.l.push(s);
    }

    // Return the vector with all the names and create a new one for this
struct
    fn return_all_and_reset(&mut self) -> Vec<String> {
        // TODO: This fails to compile, fix it!
        let temp = self.l;
        self.l = Vec::new();
        return temp
    }
}fn main() {
    let mut names = NamesList::new();
    names.add(String::from("Erik"));
    names.add(String::from("Etienne"));
    println!("Hello {:?}, it's great to see you all!",
names.return_all_and_reset());
    names.add(String::from("Rick"));
    println!("And also you {:?}", names.return_all_and_reset());
}

/* Expected output:

Hello ["Erik", "Etienne"], it's great to see you all!
And also you ["Rick"]

*/
```

NamesList is supposed to collect names (Strings) in a vector and when return_all_and_reset is called, then it's supposed to return the vector with all the names and start with a new fresh one. However it fails to compile at "let temp = self.l". Please fix the method return_all_and_reset. You are not allowed to change other methods or to change the signature of the method, but you are allowed to change the methods body and also import other functions from the Rust standard library using "use" in the header.

8 To give you some feedback about your grade, I prepared a small Rust program.

10 pt.

```
use std::io;

fn main() {
    let mut grade = String::new();

    io::stdin().read_line(&mut grade)
        .expect("Failed to read line");

    let grade: u32 = grade.trim().parse()
        .expect("Please type a number!");

    println!("{}", match grade {
        1|2|3|4|5 => "I'm so sorry",
        6 => "You made it!",
        7 => "That's good",
        8 => "That's really good",
        9 => "That's almost excellent",
        10 => "Excellent"
    });
}
```

However, the program doesn't compile. Please fix that:

9 One way how to get better results from AFL while fuzzing is to use the ASAN (Address Sanitaizer) library. Describe at least one kind of bug would you discover during fuzzing with ASAN very soon that would probably not show up while fuzzing in the default configuration of AFL without using ASAN.

8 pt.

10 A common problem for cryptographic code (and also for any other kind of programs that process secret inputs) is that the execution time might depend on the input and that an adversary might be able to learn something about the input when he is able to observe the execution time of a program processing those inputs. Commonly the difference in the execution times comes from either coditional jums which jump conditions that depend on the secret input and from memory access patterns with addresses depending on the secret input. One solution is to rewrite those programs so that the execution time doesn't depend on the secret inputs anymore.

5 pt.

Which tool can you use to verify that the rewritten program doesn't contain conditional jumps or memory access patterns that depend on the secret input anymore? We showed one such tool in the lecture and when you don't remember the name anymore, just describe roughtly what it does and how you would use and/or invoke it.

11
5 pt.

One of the (bonus) assignment was to add a security pipeline to your sorted tree C implementation project. The pipeline should compile your project and then run AFL on it for 5 minutes to find potential bugs using fuzzing.

The version of AFL running as part of your pipeline would be exactly the same as the one you could also run on your laptop computer and it would find exactly the same bugs as those you could also find just by running AFL locally. Why would you gain any advantage by running AFL as part of your security pipeline when you could also do it locally?

12
10 pt.

In more than one lecture, a slide with 10 general methods to improve the security of code was shown, which are:

- Validate input
- Listen to compiler warnings
- Design for security policies
- Keep is simple
- Default to deny
- Least privilege
- Sanitize your output
- Defence in depth
- Analyse your code
- Adopt a secure coding standard

In the last assignmed, you were asked to fix security vulnerabilities in a web application. One of those vulnerabilities was a “Cross Site Scripting (XSS)” vulnerability.

Name at least one out of those 10 methods that helps you to avoid XSS vulnerabilities in general and explain how it helps you with that.

13 8 pt. Consider the following part of the source code of the web application you examined in your last assignment:

```
// This is the handler for posting a new comment
post("/image/{imageId}/comments/post") {
    // Get the POST query
    val postParams = call.receiveParameters()
    val name =  postParams["userName"]
    val comment = postParams["comment"]

    // Get the image ID from the URL
    val imageId = call.parameters["imageId"]?.toInt()

    if (imageId == null) {
        call.respond(HttpStatusCode.NotFound)
    } else {
        // Insert the comment into the database
        conn!!
            .prepareStatement("INSERT INTO comments (image_id, user_name,
comment) VALUES ($imageId, '$name', '$comment')")
            .execute()

        // Redirect back to the image page
        call.respondRedirect("/image/$imageId")
    }
}
```

It contains at least one security vulnerability. Which vulnerability is it and how can you fix it? You are not required to provide code with your answer, just describe roughly what you would need to modify or how you would restructure the code here.