Exam Software Security, 201600051

University of Twente

1 February 2017, 13:30 - 16:30

This exam consists of 10 (sub)questions of equal weight. It is a closed-book exam: the use of any printed or online material is prohibited. Short and to-the-point answers are highly preferred over long stories.

1. The following fragments in the C-programming language illustrate typical vulnerabilities in low-level C-programming. Recall that `argc` denotes the number of command-line arguments (including the command name itself) and `argv[i]` points at the i-th argument. We use standard library functions:

```
char *strcpy(char * dst, const char * src);
char *strncpy(char * dst, const char * src, size_t len);
```

   (a) What kind of attack is possible on the following fragment? How can this be exploited? What should the programmer have written instead?

```
if (argc>1) {
  char* string = argv[1];
  printf(string);
}
```

   (b) How can the following code result in a buffer overflow?

```
if (argc>1) {
  char buffer[10];
  char dest[10];
  strncpy(buffer,argv[1],10);
  strcpy(dest,buffer);
}
```

2. Mention three different countermeasures at the platform-level (OS) against buffer overflow attacks on the stack.

3. The RUST programming language provides a safe alternative to C. The following code fragments illustrate some specific features of RUST.

   (a) The following code fragment is invalid in RUST

   ```
   let x = 4;
   x = 6;
   ```

   Explain what feature of RUST prohibits this behaviour. How should this fragment be modified?

   (b) Also the following code fragment in RUST doesn't compile. Explain why it doesn't compile in terms of a specific RUST feature.

   ```
   let mut v = vec![5,7,9];
   v.push(12);
   let v2 = v;
   println!("{}", v[2]);
   ```

4. Two ways to perform input validation are *black-listing* and *white-listing*. Discuss the main advantage and disadvantage of each method.

5. Both CSRF (Cross-site Request Forgery) and XSS (Cross-Site Scripting) depend on executing code that seems to originate from a trusted partner. What is the main difference between the two?

6. Vulnerability Detection

   (a) AFL (American Fuzzing Lop) is based on genetic programming. From each new generation of mutated inputs, it selects the best inputs. What is the used criterion to select mutants?

   (b) How could the (ideas of) Valgrind and AFL be combined to detect more vulnerability problems than AFL on its own?

   (c) How does symbolic execution help to find interesting test cases for security testing?