2021-01-20 - M-CS - Software Security - 201600051

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

Tijdsduur: 2 uur

Aantal vragen: 12

Gegenereerd op: 19 jan. 2021

Inhoud: Pagina's:

2021-01-20 - M-CS - Software Security - 201600051

Cursus: M-CS Software Security 201600051

There are 100 points in total for this exam and you have 120 minutes to complete it. That means that you should complete a question with X points in about X minutes. When you think that it will take you longer, then skip the question for now and do it at the end of the exam.

Aantal vragen: 12

In totaal zijn 101 punten voor deze toets te behalen, 55,55 punten zijn nodig om voor de toets te slagen.

1 In the lecture, we studied two different tools that almost do the same: ASAN and Valgrind.
7 pt.

2 In general, one of the main things missing in Rust compared to C and C++ are circular data
10 pt. structures. As part of one of the reflection sessions, we studied an article that describes a potential solution to this problem:

https://rust-leipzig.github.io/architecture/2016/12/20/idiomatic-trees-in-rust/

While this might provide a good solution for static data, which is just generated once but then never updated, certain problems arise when you would use that approach for dynamic data structures in which items are regularly added, modified and deleted during runtime.

Please describe what kind of problems one needs to expect here and potential solutions for those problems.

3 AFL is considered to be a “guided” fuzzer.
7 pt.

Describe what a guided fuzzer is (the difference when compared to a dumb/basic fuzzer) and how that “guidance” influences the choices AFL mades when generating new inputs.

4 In the lecture, I showed the following sample program written in Rust:
7 pt.

```
fn main() {
    let mut a = vec![1,2,3];
    a.push(4);
    let b = &a;
    println!("{}",b[2]);
    let mut c = a;
    c.push(5);
    println!("{:?}", c);
}
```

It compiles just fine with the current Rust compiler, but there were issues with older versions of the Rust compiler. Please explain what the issue with the older Rust compiler was.

5 7 pt. One of the problems of your web application was cross site scripting. One way to counter cross site scripting is to accept only certain inputs and reject others. Here are two code snippets written in Kotlin that try to filter out potentially malicious input values.

```
fun checkXss1(s: String): Boolean {
   val specialChars = arrayOf('&', '<', '>', '\'', '"')
   for (c in s) {
       if (c in specialChars) {
           return false
       }
   }
   return true
}

fun checkXss2(s: String): Boolean {
   if (s.matches("^[a-zA-Z0-9 !.,]*$".toRegex())) {
       return true
   }
   return false
}
```

Which one of them is better from a security point of view and why?

6 7 pt. Here is a short C program that has an obvious bug (out of bounds memory access).

```
#include <stdlib.h>
#define N 100

int main(int argc, char** argv) {
   char* buf[N];
   for (int i = 0; i < N; i++) {
       buf[i] = malloc(N);
       if (buf[i] == NULL) {
           return 1;
       }
   }
   buf[N-1][N] = 'a';
   return 0;
}
```

When you compile that program with address sanitization (**-fsanitize=address**) and run it, will ASAN detect the problem alway or just with a high probability or (almost) never? Explain your answer!

In case the problem is always detected, try to modify it in a way so that it's possible that the problem will not be detected by ASAN, but an out of bounds memory access is still present.

7 7 pt.

One of the approaches to support security during the development process we took a look at in the lecture was a “Security Pipeline”, which basically just means automatically running certain scripts/jobs on a server that perform various checks whenever a change is made to the application or in regular intervals.

However, there are exceptions: Describe a scenario in which such a security pipeline would probably yield in better results for a certain aspect in security than a human expert who would inspect the program regularly.

8 7 pt.

Malloc returns a pointer towards the newly allocated memory. Of course a programmer has to take this pointer and use it somehow. Discarding the return value of malloc is obviously a bug. It can easily be detected with for example ASAN or Valgrind as long as the bug is triggered during the execution of the program, but when the relevant part of the program that contains this bug is not triggered by one of the test inputs, it will not be found by Valgrind or ASAN.

Which tool can you detect this problem anyway without having to somehow find an input for the program that triggers the bug? Why do you think that this tool will find the bug and how will it do that?

9

One of your tasks was to write a content security policy for your application. Such an CSP might look like that.

```
script-src 'self' https://code.jquery.com/jquery-3.3.1.slim.min.js
https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js
https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js;
style-src 'self' https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/
bootstrap.min.css;
img-src 'self';
form-action 'self';
```

10 pt. **a.** How can you modify the policy to make it more robust/secure without altering the application itself?

1 pt. **b.** What kind of changes could you make to the application itself that would allow a more restrictive content security policy?

10 10 pt.

One of the methods to detect bugs in a program is fuzzing. We discussed it in the lecture and we used AFL as an example for a guided fuzzer. AFL needs a sample input for a program called seed so that it can start fuzzing. For example, when we would like to test our sorted tree C implementation, we could use the following seed:

```
i 1 erik
c 1 erik
p
e 1 erik
c 1 erik
q
```

Assume that we would now like to introduce an additional feature for the sorted tree, namely authentication: A password needs to be supplied as the first line of input. The application reads the correct password from ~/.config/sorted_tree_password and compares it to the password supplied via the standard input. For example when the password “SoftwareSecurityIsGreat” is configured in ~/.config/sorted_tree_password, then the following would be a valid input file:

```
SoftwareSecurityIsGreat
i 1 erik
i 2 tews
p
q
```

But an input like:

```
Test1234
i 1 erik
i 2 tews
p
q
```

Would be directly rejected after the first line of input. Obviously, AFL would not be able to immediately guess the right password, so starting from the current seed would probably not result in many different paths discovered.

Please sketch two different approaches how you could still fuzz the sorted tree with AFL, potentially using some additional tools that assist AFL.

11 14 pt. Implementing erase for your sorted tree in Rust was (besides balancing) probably the most difficult part of your implementation task. Here, you see an attempt to implement a sorted tree in Rust with an erase function.

```
use std::cmp::Ordering;

#[derive(PartialEq, Eq, Hash, PartialOrd, Ord)]
struct Data{
   name : String,
   age : i32
}

struct Node{
   data: Data,
   left : Option<Box<Node>>,
   right: Option<Box<Node>>,
}

fn insert_into_right(node : &mut Box<Node>, to_insert : Box<Node>){
   match &mut node.right {
       None => {node.right = Some(to_insert)},
       Some(r) => {insert_into_right(r, to_insert)}
   }
}

fn erase_node(input_node : &mut Option<Box<Node>>, data: Data) -> bool {
   match input_node {
       None => {false},
       Some(node) => {
           match data.cmp(&node.data) {
               Ordering::Less => {erase_node(&mut node.left, data)},
               Ordering::Greater => {erase_node(&mut node.right, data)},
               Ordering::Equal => {
                   match (node.left, node.right) {
                       (None,None)              => {*input_node = None;
true},
                       (left,None)              => {*input_node = left;
true},
                       (None,right)             => {*input_node = right;
true},
                       (Some(mut left),Some(right)) => {
                           insert_into_right(&mut left,right);
                           *input_node = Some(left);
                           true}
                   }
               }
           }
       }
```

```
    }
}

fn main() {
    println!("Hello, world!");
}
```

Please fix that issue.

12 7 pt. One of the common problems of web applications is using components that are outdated and have known vulnerabilities. For example the image gallery used the *com.fasterxml.jackson* Java library for XML parsing that had a known vulnerability in the version that was included by the image gallery. However other Java/Kotlin libraries were not the only kind of dependency the image gallery had.

Which two other kinds of dependencies had your application?

Naam:				Handtekening:
Datum:	- -	Geboortedatum:	- -	

Cursus: M-CS-201600051-1B M-CS Software Security 201600051 - Vragen: 2021-01-20 - M-CS - Software Security - 201600051

There are 100 points in total for this exam and you have 120 minutes to complete it. That means that you should complete a question with X points in about X minutes. When you think that it will take you longer, then skip the question for now and do it at the end of the exam.

1
7 pt.

Antwoord:

2
10 pt.

Antwoord:

2

4

6

8

10

12

14

3
7 pt.

Antwoord:

4
7 pt.

Antwoord:

2

4

6

8

10

12

14

5
7 pt.

Antwoord:

6
7 pt.

Antwoord:

2

4

6

8

10

12

14

7
7 pt.

Antwoord:

8
7 pt.

Antwoord:

9
11 pt. **a.**

Antwoord:

2

4

6

8

10

12

14

b. Antwoord:

2

4

6

8

10

12

14

10
10 pt.

Antwoord:

2

4

6

8

10

12

14

11
14 pt.

Antwoord:

2

4

6

8

10

12

14

16

18

20

22

24

26

28

30

12
7 pt.

Antwoord:

2

4

6

8

10

12

14

Correctiemodel

1.
7 pt.

Correction criterion	Points
Criterion 1	7 points
Total points:	*7 points*

2.
10 pt.

Correction criterion	Points
Criterion 1	10 points
Total points:	*10 points*

3.
7 pt.

Correction criterion	Points
Criterion 1	7 points
Total points:	*7 points*

4.
7 pt.

Correction criterion	Points
Criterion 1	7 points
Total points:	*7 points*

5.
7 pt.

Correction criterion	Points
Criterion 1	7 points
Total points:	*7 points*

6.
7 pt.

Correction criterion	Points
Criterion 1	7 points
Total points:	*7 points*

7.
7 pt.

Correction criterion	Points
Criterion 1	7 points
Total points:	*7 points*

8.
7 pt.

Correction criterion	Points
Criterion 1	7 points
Total points:	*7 points*

9.
11 pt.

a.

Correction criterion	Points
Criterion 1	10 points
Total points:	*10 points*

b.

Correction criterion	Points
Criterion 1	1 point
Total points:	*1 point*

10.
10 pt.

Correction criterion	Points
Criterion 1	10 points
Total points:	*10 points*

11.
14 pt.

Correction criterion	Points
Criterion 1	14 points
Total points:	*14 points*

12.
7 pt.

Correction criterion	Points
Criterion 1	7 points
Total points:	*7 points*

Cesuur

Toegepaste raadscore: 0 pt

Behaalde punten	Cijfer
101	10
100	9,9
99	9,8
98	9,7
97	9,6
96	9,5
95	9,4
94	9,3
93	9,2
92	9,1
91	9,0
90	8,9
89	8,8
88	8,7
87	8,6
86	8,5
85	8,4
84	8,3
83	8,2
82	8,1
81	8,0
80	7,9
79	7,8
78	7,7
77	7,6
76	7,5
75	7,4
74	7,3
73	7,2
72	7,1
71	7,0
70	6,9
69	6,8
68	6,7
67	6,6
66	6,5
65	6,4
64	6,3
63	6,2
62	6,1
61	6,0
60	5,9
59	5,8
58	5,7
57	5,6
56	5,5
55	5,5
54	5,4
53	5,3
52	5,2
51	5,1
50	5,1
49	5,0
48	4,9
47	4,8
46	4,7
45	4,6
44	4,6
43	4,5

42	4,4
41	4,3
40	4,2
39	4,2
38	4,1
37	4,0
36	3,9
35	3,8
34	3,8
33	3,7
32	3,6
31	3,5
30	3,4
29	3,3
28	3,3
27	3,2
26	3,1
25	3,0
24	2,9
23	2,9
22	2,8
21	2,7
20	2,6
19	2,5
18	2,5
17	2,4
16	2,3
15	2,2
14	2,1
13	2,1
12	2,0
11	1,9
10	1,8

9	1,7
8	1,6
7	1,6
6	1,5
5	1,4
4	1,3
3	1,2
2	1,2
1	1,1
0	1,0

Vraag-identificatiecodes

Deze identifiers kunnen worden gebruikt om de precieze vraag in de vragenbanken te identificeren. Gebruik deze code in combinatie met de documentcode wanneer u feedback doorgeeft, zodat precies duidelijk is op welke vraag en -versie uw feedback van toepassing is.

Documentidentificatiecode: 4390-5205

Vraagnummer	Vraag-identificatiecode	Versie-identificatiecode
1	47460	662532c1-6b77-fd98-3174-cf6b735ea44b
2	47455	d6938e8a-6b72-0c4e-e348-46b12b40006f
3	47475	29eb2626-8ede-159c-9ffe-13265d0251d1
4	47450	c8dd8570-ba68-0eb0-1151-0fef5004ace2
5	47435	24e50aae-c27e-082e-7262-2f70a3ce5544
6	47440	6d677e95-ceca-720b-3f49-3b1d6e248e6b
7	47470	954553ff-8ecc-96a6-bdb3-215f59ad70dc
8	47480	a2d4b0be-e583-7324-d4f6-81883e58fe4e
9	47430	7156de84-2d06-945f-006e-728ffc08c793
10	47445	6e0bd353-8a77-96e0-5269-a1d22c0f3204
11	47465	9d2b615a-5ad8-cb42-b74c-f2c5b7ce79ae
12	47425	f252072d-1432-fb8d-92bc-6166cc30584d