Vraag 1

Beantwoord op: 29 januari 2025 - 08:54 **Duration:** 10 min. en 14 sec. **Score:** 5 van 10 pt. **5 pt.**

Consider the following code snippet:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define BUFFER_SIZE 10

void processInput(char* input) {
    // This function adds a prefix to the input
    char prefix[] = "Data: ";
    char* combinedInput = (char*)malloc(strlen(prefix) + strlen(input) + 1);

    // Concatenate the prefix and input
    strcpy(combinedInput, prefix);
    strcat(combinedInput, input);

    printf("Combined input: %s\n", combinedInput);
}

void copyInput(char* userInput) {
    char buffer[BUFFER_SIZE];
    strcpy(buffer, userInput);
    printf("Buffer content: %s\n", buffer);
}

int main() {
    char input[50];

    printf("Enter some text: ");
    fgets(input, sizeof(input), stdin);
    // Remove the newline character from fgets
    input[strcspn(input, "\n")] = 0;

    // Process input before passing it to the other function
    processInput(input);
    copyInput(input);
    return 0;
}
```

Identify two memory-related vulnerabilities in this code snippet. (4pts)

The memory allocated by malloc in processInput is never freed, possibly causing memoryleaks and strcpy and strcat are used instead of their more secure countervarients strlnpy and strlnat, thus we do not know if the copied strings are null-terminated, possibly causing out-of-bouds access

2 pt.

Status: Klaar met nakijken

1. Buffer overflow. Memory not freed
   2 van 4 pt.

Explain the role of dynamic memory allocation in posing security risks. Describe at least two risks. (2pts)

With dynamic memory allocation more freedom is given than with its static variant. But as with programming, more freedom also means that there are more possible mistakes. 2 of these mistakes or risks that occur due to dynamic memory management are overwriting initialized memory and reading unintended memory blocks

1 pt.

Status: Klaar met nakijken

1. Memory leaks; Double free(); Buffer overflow; Memory allocated with malloc is not initialized
   1 van 2 pt.

Propose appropriate fixes to the code snippet so it is no longer vulnerable. (4pts)

replace every strcpy and strcat by strlnpy and strlnat and free all memory at the end of main

2 pt.

Status: Klaar met nakijken

1. Check if combinedInput is not null and that malloc() actually worked. Replace strcpy and strcat with snprintf. Free combinedInput to make sure no memory leak
   2 van 4 pt.

Vraag 2

Beantwoord op: 29 januari 2025 - 08:59 **Duration:** 9 min. en 35 sec. **Score:** 8 van 10 pt. **8 pt.**

Consider the following code snippet:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void regularbeats(char *payload, int payload_length) {
    char buffer[64];
    memcpy(buffer, payload, payload_length);
    printf("Response: %s\n", buffer);
}

int main() {
    char *userInput = "Hello, this is a test message.";
    int length = 37;
    // Call the handler
    regularbeats(userInput, length);
    return 0;
}
```

Identify the location of the vulnerability in this code. (2pts)

memcpy

2 pt.

Status: Klaar met nakijken

1. payload length and the size of payload do not match.
   2 van 2 pt.

What is the name by which this vulnerability is known? (2pts)

buffer overflow

0 pt.

Status: Klaar met nakijken

1. heartbleed
   0 van 2 pt.

To what end can an attacker exploit this vulnerability? (6pts)

Because of the way that the memcpy is set up, it is possible to copy more (or less) that just the intended user input. This can lead to buffer overflows and can be used to access memory that should not be accessed through this function

6 pt.

Status: Klaar met nakijken

1. can read potentially sensitive data from out of bounds memory locations
   6 van 6 pt.

Vraag 3

Beantwoord op: 29 januari 2025 - 09:52 **Duration:** 4 min. en 13 sec. **Score:** 10 van 10 pt. **10 pt.**

Consider the following Content Security Policy:

```
Content-Security-Policy: img-src https://trusted-images.com; script-src *;
```

What are the security risks faced by the web application if this CSP is used? (5pts)

For scripts, all sources are allowed and unchecked. This means that malicious scripts can be inserted in the web application. (XSS)
This also goes for all other undefined sources, but in this CSP, the script-src it is even explicitely allowed.

Aantal woorden: 40, aantal tekens: 243

5 pt.

Status: Klaar met nakijken

1. Scripts alllowed from everywhere. Risk of injection
   5 van 5 pt.

Modify the CSP so that it is secure/robust to security threat(s) mentioned above. (5pts)

```
Content-Security-Policy: img-src https://trusted-images.com; script-src self; (add more relevand source
```

Aantal woorden: 13, aantal tekens: 132

5 pt.

Status: Klaar met nakijken

1. Content-Security-Policy: img-src https://trusted-images.com;
   script-src https://trusted-images.com;
   5 van 5 pt.

Vraag 4

Beantwoord op: 29 januari 2025 - 10:06 **Duration:** 16 min. en 10 sec. **Score:** 10 van 10 pt. **10 pt.**

Provide a definition for:

- Common Vulnerabilities and Exposures (CVE) (1pt)
- Common Weakness Enumeration (CWE) (1pt)
- Common Vulnerability Scoring System (CVSS) (1pt)

CVE: a specific bug or vulnerability in a specific program
CWE: category of vulnerabilities, f.e. buffer overflow
CVSS: a scoring system for the impact of a bug

Aantal woorden: 27, aantal tekens: 160

3 pt.

Status: Klaar met nakijken

1. CWE abstract weakness
   1 van 1 pt.
2. CVE concrete vuln found in a software version
   1 van 1 pt.
3. CVSS severity score assigned to CVE
   1 van 1 pt.

CWE's are categories, CVE's are specific bugs. Every CVE belongs to a CWE. f.e. if the CWE is buffer-overflow then a CVE that belongs to the CWE might be a specific buffer overflow bug in microsoft word. (lets call this bug CVE-2019-119)
CVSS scores CVE impacts on a 1-10 scale. So for example, CVE-2019-119 has a CVSS impact of 7.5.
With the CVSS a ranking of CWE's can be made -> amount of CVE's * average CVSS of a CVE belonging to the CWE.

Example from OWASP Top-10:
So for example, say that we have the CWE: cryptographic failure, in this CWE, we have the CVE: non-encrypted meta-data on whatsapp web and this CVE gets a CVSS score of 8.0. This CVSS combined with other cryptographic failures result in the CWE: cryptographic failure to get a average CVSS of 7.5

Aantal woorden: 140, aantal tekens: 768

7 pt.

Status: Klaar met nakijken

1. CWE - parent of many CVEs
   2 van 2 pt.
2. CVSS - assigned to a CVE
   2 van 2 pt.
3. Example: injection. CWE-XSS. CVE-specific vulnerability CVE-2024-001. CVSS-10
   3 van 3 pt.

Vraag 5

Beantwoord op: 29 januari 2025 - 09:16 **Duration:** 7 min. en 3 sec. **Score:** 2 van 6 pt. **2 pt.**

Please indicate which of the following code snippets will be successfully compiled by the Rust compiler (if compiled in isolation, as part of the main() function), and which ones will not. (6pts)

Rust:
```
let x: i64 = 123;
let mut y: u32 = 90;
y = 156;
y = x;
println!("{}", x);
```

- ○ Compiles
- → ◉ Does not compile 1 pt. ✓

1 pt.

Rust:
```
let a = 10;
let b = a;
println!("{}", a);
```

- ○ Compiles 1 pt. ✓
- → ◉ Does not compile ✗

0 pt.

Rust:

```
let a = 10;
let b = 8;
let c = 7;
b = c + a;
println!("{}", a);
```

- ○ Compiles
- ◉ Does not compile — 1 pt. ✔

1 pt.

Rust:

```
let x = "abc";
let y = x;
println!("{}", x);
println!("{}", y);
```

- ○ Compiles — 1 pt. ✔
- ◉ Does not compile ✘

0 pt.

Rust:

```
let x = String::from("abc");
let y = x;
println!("{}", y);
```

- ◉ Compiles ✘
- ○ Does not compile — 1 pt. ✔

0 pt.

Rust:

```
let es = vec![4,5,6,7];
for i in &es {
    println!("{}", i);
}
es.push(9);
```

- ◉ Compiles ✘
- ○ Does not compile — 1 pt. ✔

0 pt.

Vraag 6

Beantwoord op: 29 januari 2025 - 10:26 **Duration:** 37 min. en 35 sec. **Score:** 4 van 24 pt. **4 pt.**

Consider the following incomplete program:

Rust:

```
fn aaa(bbb: BBB, ddd: CCC) -> CCC {
    CCC::D(bbb, Box::new(ddd))
}

fn fff(bbb: BBB, ccc: CCC) -> CCC {
    match ccc {
        CCC::E => return aaa(bbb, CCC::E),
        CCC::D(eee, ddd) => return aaa(eee, fff(bbb, *ddd))
    }
}
```

Try to get an intuition about how it might work before proceeding. Now, you need to make it compilable. In order to do that, you'll need to tackle the following points:

Declare the datatype CCC, based on how the functions deal with values of type CCC. (8pts)

2 pt.

struct CCC {

E: int

D: Box

}

Aantal woorden: 8, aantal tekens: 39

Status: Klaar met nakijken

1. pub enum CCC {
   E,
   D(BBB, Box<CCC>)
   }
   2 van 8 pt.

Explain which types could be valid for BBB. Also, explain and provide some examples of types that would NOT be valid. (8pts)

0 pt.

It seems that the BBB type is some sort of iterable datatype, we see it used as a first argument with some type of invoke to a datastructure, which sometimes is completely new, suggesting it is data to be used in the structure.
valid examples: integer, string
not valid examples: Result, Option

Aantal woorden: 52, aantal tekens: 294

Status: Klaar met nakijken

1. i32 works, as does any other predefined numeric type, char, bool, immutable arrays of these, or \&str. More generally, any type implementing the `Copy` trait works
   0 van 4 pt.
2. does not work for references, mutable or not (unless the definition of `CCC` includes a lifetime parameter associated with the `BBB` element of its `D` case), nor does it work with any type implementing trait `Drop`, as this excludes the implementation of trait `Copy`. Examples of such types include `String`, `Vec`, and `Box`.
   0 van 4 pt.

Function `aaa()' makes use of the `Box' type defined by Rust. If we stop using that type (and its associated functions, e.g., `new()') in this program, i.e., we simply remove it everywhere it appears, it will stop compiling.

Explain why this happens and why we actually need to use `Box' in this example. (8pts)

2 pt.

If we remove the box-type, the code stops compiling because the structure necessary for CCC to work gets taken away.
We need the Box type because we are itiratively using CCC within CCC

Aantal woorden: 33, aantal tekens: 185

Status: Klaar met nakijken

1. because the data structure type is recursively defined so Rust will not know how much memory will need to be allocated in advance, which Rust does not allow for stack-allocated data.
   2 van 8 pt.

Vraag 7

Beantwoord op: 29 januari 2025 - 09:39 **Duration:** 11 min. en 33 sec. **Score:** 2 van 10 pt. **2 pt.**

Consider the following code:

c:
```c
struct object{
    int id;
    char code;
};

void ocopy(struct object * dst, struct object * src) {
    *dst= *src;
}

int main(int argc, char** argv) {
    struct object a;
    a.id = 20;
    a.code= 'c';
    struct object b;
    ocopy(&b, &a);
    return 0;
}
```

What is the size of the structure `object' in bytes? (3pts)

size int + size char
with default sized for integers and a character

Aantal woorden: 13, aantal tekens: 68

2 pt.

Status: Klaar met nakijken

1. int is typically 4 bytes and char is 1 byte, so 5 bytes
   2 van 3 pt.

How many bytes will a typical malloc() operation allocate for the structure `object'? (3pts)

size int + size char + 1
the size of the structure + the null-byte

Aantal woorden: 15, aantal tekens: 66

0 pt.

Status: Klaar met nakijken

1. with padding of 3 bytes, 8 bytes total
   0 van 3 pt.

When compiled with address and memory sanitizer, what type of warnings will they report? (2pts + 2pts)

Malloc is never called, thus only dynamic memory allocation is used, causing possible undefined behaviour. the object that gets copied is not freed afterwards, possibly resulting in memory leaks later on

Aantal woorden: 31, aantal tekens: 203

0 pt.

Status: Klaar met nakijken

1. None from ASAN
   0 van 2 pt.
2. MSAN will likely flag the copying of uninitialized padding bytes from src to dst.
   0 van 2 pt.

Vraag 8

Beantwoord op: 29 januari 2025 - 09:45 **Duration:** 5 min. en 47 sec. **Score:** 10 van 10 pt. **10 pt.**

The information security triad is defined by Confidentiality, Integrity, and Availability.

When a code snippet has a memory leak due to missing free() statements, which of the above mentioned properties are affected and why? (5pts)

possibly all
confidentiality: because of the missing free statement, the memory is kept and could possibly be accessed later
integrity: missing free statements can result in memory being overwritten, compromising the integrity
availability: Because memory is not freed, free memory could run out and the program would struggle with memory availibility

Aantal woorden: 51, aantal tekens: 351

5 pt.

Status: Klaar met nakijken

1. Availability will be impacted by using up all free memory
   5 van 5 pt.

How could an attacker exploit a memory leak to compromise the confidentiality of the system? (5pts)

An attacker could write an exploit that unvoluntairily froces the program to access different memory than intended. Memory that should not be accessed.

Aantal woorden: 23, aantal tekens: 151

5 pt.

Status: Klaar met nakijken

1. Access to memory out of bounds could mean you're reading sensitive content from other processes
   5 van 5 pt.

Vraag 9

Beantwoord op: 29 januari 2025 - 09:51 **Duration:** 6 min. en 7 sec. **Score:** 6.5 van 10 pt. **6,5 pt.**

AFL is considered to be a mutation-based guided fuzzer.

Describe what a guided fuzzer is (how it differs from a basic fuzzer) (5pts)

In a guided fuzzer the fuzzer uses a metric or help from the user to determine interesting inputs instead of brute-force trying everything like a bacis fuzzer would.

Aantal woorden: 28, aantal tekens: 165

2,5 pt.

Status: Klaar met nakijken

1. Coverage-guided fuzzer and it uses code coverage for feedback loop
   2,5 van 5 pt.

How does the guidance influence the choices AFL makes when generating new test inputs? (5pts)

In AFL's case the guidance is based on coverage, which means that AFL with prioritize paths that result in more coverage in favor of those that yield no new results. meaning that test inputs that have resulted in more coverage are used to try and attempt to find even more paths.

Aantal woorden: 51, aantal tekens: 279

4 pt.

Status: Klaar met nakijken

1. instrumentation used to monitor branches. AFL mutates inputs covering previously uncovered program behavior. Helps prioritize inputs that explore untested code regions
   4 van 5 pt.