# Project 5: Secure Coding. Input validation and static analysis

- Namruth Reddy
- suryanarayanareddy.n@northeastern.edu
- NUID 001523553

Output of the randomizer script and files obtained to work with

```
namruth@DESKTOP-LETJM2M:/mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_project5/securecoding$ ./randomizer.sh 0015
23553 suryanarayanareddy.n@northeastern.edu
Thanks, your email is: suryanarayanareddy.n@northeastern.edu,
and your 9 programs to work with are:
1. codeN1108.c
2. codeN1110.c
3. codeN1112.c
4. codeN1114.c
5. codeN1116.c
6. codeN1118.c
7. codeN1120.c
8. codeN1122.c
9. codeN1124.c
namruth@DESKTOP-LETJM2M:/mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_project5/securecoding$
```

## Part 1: Validate input of the randomizer script using Regex

Extra code added to validate NUID, and NU email id is shown in the picture below:

```
#!/bin/bash
if [[ $1 =~ ^00[0-9]{7}$ ]]
then
  echo "NUID valid"
else
  echo "NUID invalid, exitting the program"
  exit 0
fi

if [[ $2 =~ ^[A-Za-z]+\.[A-Za-z0-9]+@northeastern.edu$ ]]
then
  echo "NU email valid"
else
  echo "NU email invalid, exitting the program"
  exit 0
fi
```

$1 is the first input which is NUID, and the regex pattern uses the following rules:

- NUID starts with 00. ^ is used to show the starting characters.
- Then it's followed by any number between 0-9 (7 times). Ends with this as well. $ is used for that

$2 is the NU email id and the regex pattern uses the following rules:

- Email starts with surname. (It can be caps or small letters; both are valid email addresses). + indicates 1 or more characters. ^ is used to show the starting characters.
- Followed by a . (dot). \ is used as an escape character.
- Then it is followed by a letter or sometimes group of letters can include numbers here. + again for 1 or more characters.
- Email always ends with @northeastern.edu. $ is used for that.

# Part 2 and Part3: Coding, Running the Analysis and Mapping to CWE.

# CodeN1108.c

**Output of clang static analyzer report**

## securecoding - scan-build results

| | |
|---|---|
| **User:** | namruth@DESKTOP-LETJM2M |
| **Working Directory:** | /mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_project5/securecoding |
| **Command Line:** | gcc codeN1108.c |
| **Clang Version:** | clang version 10.0.0-4ubuntu1 |
| **Date:** | Sat Apr 9 16:43:26 2022 |

### Bug Summary

| Bug Type | Quantity | Display? |
|---|---|---|
| **All Bugs** | **1** | ☑ |
| **Logic error** | | |
| Uninitialized argument value | 1 | ☑ |

### Reports

| Bug Group | Bug Type ▾ | File | Function/Method | Line | Path Length | |
|---|---|---|---|---|---|---|
| Logic error | Uninitialized argument value | codeN1108.c | main | 30 | 12 | View Report |

**Mapping the bug to CWE known weaknesses and how to fix it**

This bug can be attributed to CWE-457: Use of Uninitialized Variable. More details about it can be found here - https://cwe.mitre.org/data/definitions/457.html

This weakness can be fixed by having the right initialization value and then checking the values before accessing or printing them

**Fixing the clang errors and output of the program obtained**

**This is the fixed code:**

```c
#include <stdio.h>

struct SoccerTeam {
    char name[50];
    int revenueThisYear;
    long revenueOverall;
};

struct SoccerTeam arsenalFC = {.name = "Arsenal", .revenueOverall = 12300000, .revenueThisYear = 2300000};
struct SoccerTeam barcelonaFC = {.name = "Barcelona", .revenueOverall = 1220000, .revenueThisYear = 9900000};
struct SoccerTeam parisSG = {.name = "Paris St", .revenueOverall = 1220000, .revenueThisYear = 55200000};

long calculateSoccerRevenue(long revenueTillDate, long revenueThisYear) {
    return revenueTillDate + revenueThisYear;
}

int main() {

    struct SoccerTeam teams[] = {arsenalFC, barcelonaFC, parisSG};
    long maxRevenue = 0;
    char *maxRevenueTeam = NULL;
    for (int i = 0; i < 3; i++) {
        long totalRevenue = calculateSoccerRevenue(teams[i].revenueOverall, teams[i].revenueThisYear);
        printf("%s: %ld\n", teams[i].name, totalRevenue);
        if (totalRevenue > maxRevenue) {
            maxRevenue = totalRevenue;
            maxRevenueTeam = teams[i].name;
        }
    }
    if (maxRevenue && maxRevenueTeam)
        printf("Team with max revenue (%ld) is: %s\n", maxRevenue, maxRevenueTeam);

    return (0);
}
```

**And the output obtained:**

```
namruth:securecoding$ scan-build -o . gcc codeN1108.c
scan-build: Using '/usr/lib/llvm-10/bin/clang' for static analysis
scan-build: Removing directory '/mnt/c/Users/namru/OneDrive/Documents/CSS
022-04-09-165423-20388-1' because it contains no reports.
scan-build: No bugs found.
namruth:securecoding$ ./a.out
Arsenal: 14600000
Barcelona: 11120000
Paris St: 56420000
Team with max revenue (56420000) is: Paris St
namruth:securecoding$
```

**Changes made to the code**

- Initializing maxRevenueTeam to NULL and then checking for it before its printed. Line numbers 21 and 30
- Changing return type of CalculateSoccerRevenue to long because long + long should be equivalent to long. Line number 13.
- Changing maxRevenue to long because short is not enough to support the revenues from the clubs. Line number 20.

# CodeN1110.c

**Output of clang static analyzer report**

No bugs were found by clang for this code. See the screenshot below.

```
namruth:securecoding$ scan-build -o . gcc codeN1110.c
scan-build: Using '/usr/lib/llvm-10/bin/clang' for static analysis
scan-build: Removing directory '/mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding
04-10-081808-560-1' because it contains no reports.
scan-build: No bugs found.
namruth:securecoding$
```

There are no security vulnerabilities in this code as such but some of the defensive protections that can be used are shown below:

```
4   #define MAX_PASSWORD_LEN 20
5   #define MAX_USERNAME_LEN 20
6
7   struct User {
8       char username[MAX_USERNAME_LEN];
9       char password[MAX_PASSWORD_LEN];
10      int uid;
11  };
```

```
75  int main(int argc, char **argv) {
76      char username[MAX_USERNAME_LEN];
77      char password[MAX_PASSWORD_LEN];
78      printf("Enter username:");
79      scanf("%19s", username);
80      printf("Enter password:");
81      scanf("%19s", password);
82      checkCredentials(username, password);
83  }
```

Using macros for max username length and max password length throughout the program for arrays initialization. Line numbers 4,5,8,9,76,77

```
41  int checkValidUserName(char *username) {
42      for (int i = 0; i < 3; i++) {
43          if (strncmp(username, usersDb[i].username,MAX_USERNAME_LEN) == 0) {
44              return 1;
45          }
46      }
47      return 0;
    }
50  int checkValidPassword(char *password) {
51      for (int i = 0; i < 3; i++) {
52          if (strncmp(password, decrypt(usersDb[i].password, strlen(usersDb[i].password)), MAX_PASSWORD_LEN) == 0) {
53              return 1;
54          }
55      }
56      return 0;
57  }
58
```

Using strncmp instead of strcmp to avoid buffer overread and max read limit can be set to macros define above. Line numbers 43, 52.

```
49
50  int checkValidPassword(char *password) {
51      for (int i = 0; i < 3; i++) {
52          if (strncmp(password, decrypt(usersDb[i].password, strlen(usersDb[i].password)), MAX_PASSWORD_LEN) == 0) {
53              return 1;
54          }
55      }
56      return 0;
57  }
58
59
60  int checkCredentials(char *username, char *password) {
61      if (checkValidUserName(username)) {
62          if (checkValidPassword(password)) {
63              printf("Welcome %s\n", username);
64              return 1;
65          } else {
66              printf("%s", "Password is invalid\n");
67              return 0;
68          }
69      } else {
70          printf("%s", "Username is invalid\n");
71      }
72      return 1;
73  }
```

Modifying checkValidPassword function to only check for password because we already have a function that checks for username. Line numbers 50,52, 62.

**Output obtained after code changes**

```
namruth:securecoding$ scan-build -o . gcc codeN1110.c
scan-build: Using '/usr/lib/llvm-10/bin/clang' for static analysis
scan-build: Removing directory '/mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_pr
022-04-10-082653-587-1' because it contains no reports.
scan-build: No bugs found.
namruth:securecoding$ ./a.out
Enter username:admin
Enter password:admin
Welcome admin
namruth:securecoding$
```

**Mapping the bug to CWE known weaknesses and how to fix it**

This code file can be attributed CWE-321: Use of Hard-coded Cryptographic Key. More details about it can be found here - https://cwe.mitre.org/data/definitions/321.html

This weakness can be fixed by not having the cryptographic key hardcoded in the program. Line number 19. This involves changing the design of the program and having the secret key stored in a separate file and retrieved only by the program with the right access.

# CodeN1112.c

**Output of clang static analyzer report**

## securecoding - scan-build results

| | |
|---|---|
| **User:** | namruth@DESKTOP-LETJM2M |
| **Working Directory:** | /mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_project5/securecoding |
| **Command Line:** | gcc codeN1112.c |
| **Clang Version:** | clang version 10.0.0-4ubuntu1 |
| **Date:** | Sun Apr 10 06:53:24 2022 |

### Bug Summary

| Bug Type | Quantity | Display? |
|---|---|---|
| All Bugs | 1 | ☑ |
| **Logic error** | | |
| Dereference of null pointer | 1 | ☑ |

### Reports

| Bug Group | Bug Type ▾ | File | Function/Method | Line | Path Length | |
|---|---|---|---|---|---|---|
| Logic error | Dereference of null pointer | codeN1112.c | method | 13 | 4 | View Report |

**Mapping the bug to CWE known weaknesses and how to fix it**

This bug can be attributed CWE-476: NULL Pointer Dereference. More details about it can be found here - https://cwe.mitre.org/data/definitions/476.html

This weakness can be fixed by having proper initialization and checking for pointers if they are NULL before dereferencing them.

**Fixing the clang errors and output of the program obtained**

**This is the fixed code:**

```c

// custom method to print digits from 11 to value of size in main().
void method(int *rtp, int *rtp1) {
    int count = 0;
    for (int i = 0; i < (*rtp1); i++) {
        printf("Numbers are: %d \n", (*rtp + count));
        count++;
    }

}

// custom method to fill an array with numbers till valid defined "macro' size
void memory_filler(int arr[]) {
    int inc = 0;
    for (int k = 0; k < ARRAY_MEMORY_SIZE; k++) {
        arr[k] = inc;
        inc += 1;
        printf("Array element is: %d \n", arr[k]);
    }

}


int main() {
    // variable initializations

    int first_number = 10;
    int size = 9;
    int random_array[ARRAY_MEMORY_SIZE];

    //Note for developers: use as size in method using defined pointers (not directly)

    int *ptr_to_first_number = &first_number;
    int *ptr_to_size = &size;
    // Hint:: add the assign the references to the pointers here:

```

**And the output obtained:**

```
namruth:securecoding$ scan-build -o . gcc codeN1112.c
scan-build: Using '/usr/lib/llvm-10/bin/clang' for static analysis
scan-build: Removing directory '/mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_proj
022-04-10-070128-312-1' because it contains no reports.
scan-build: No bugs found.
namruth:securecoding$ ./a.out
Numbers are: 10
Numbers are: 11
Numbers are: 12
Numbers are: 13
Numbers are: 14
Numbers are: 15
Numbers are: 16
Numbers are: 17
Numbers are: 18
Array element is: 0
Array element is: 1
Array element is: 2
Array element is: 3
Array element is: 4
Array element is: 5
Array element is: 6
Array element is: 7
Array element is: 8
Array element is: 9
namruth:securecoding$
```

**Changes made to the code**

- Ptr_to_first_number is initialized to address of first number and ptr_to_size is initialized to address of size. Line number 41 and 42
- Count variable is initialized to 0 to avoid garbage values. Line number 12
- For loop is limited to ARRAY_MEMORY_SIZE to avoid buffer overflow. Line number 23

# CodeN1114.c

**Output of clang static analyzer report**

## securecoding - scan-build results

| | |
|---|---|
| **User:** | namruth@DESKTOP-LETJM2M |
| **Working Directory:** | /mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_project5/securecoding |
| **Command Line:** | gcc codeN1114.c |
| **Clang Version:** | clang version 10.0.0-4ubuntu1 |
| **Date:** | Sun Apr 10 07:08:10 2022 |

### Bug Summary

| Bug Type | Quantity | Display? |
|---|---|---|
| **All Bugs** | **1** | ☑ |
| **Logic error** | | |
| Dereference of null pointer | 1 | ☑ |

### Reports

| Bug Group | Bug Type ▾ | File | Function/Method | Line | Path Length | |
|---|---|---|---|---|---|---|
| Logic error | Dereference of null pointer | codeN1114.c | swap | 12 | 4 | View Report |

**Mapping the bug to CWE known weaknesses and how to fix it**

This bug can be attributed CWE-476: NULL Pointer Dereference. More details about it can be found here - https://cwe.mitre.org/data/definitions/476.html

This weakness can be fixed by having proper initialization and checking for pointers if they are NULL before dereferencing them.

**Fixing the clang errors and output of the program obtained**

**This is the fixed code:**

```c
// custom method to swap digits (without using third variable/pointer) passed to it as arguments from main()
void swap(int *ptr1, int *ptr2) {

   if (!ptr1 || !ptr2){
     printf("One of the pointers passed is NULL, cant be swapped\n");
     return;
   }

    printf("Original First Number is : %d \n ", (*ptr1));
    printf("Original Second Number is : %d \n ", (*ptr2));

    // logic for swapping
    *ptr1 = *ptr1 + *ptr2;
    *ptr2 = *ptr1 - *ptr2;
    *ptr1 = *ptr1 - *ptr2;

    printf("Swapped First Number is : %d \n", (*ptr1));
    printf("Swapped Second Number is : %d \n", (*ptr2));

}

int main() {
    int first_number = 10;
    int second_number = 20;
    //Note for developers(students): use numbers in method using defined pointers (not directly)

    int *ptr_to_first_number = &first_number;
    int *ptr_to_second_number = &second_number;
    // Hint:: add the assign the references to the pointers here:

```

**And the output obtained:**

```
namruth:securecoding$ scan-build -o . gcc codeN1114.c
scan-build: Using '/usr/lib/llvm-10/bin/clang' for static analysis
scan-build: Removing directory '/mnt/c/Users/namru/OneDrive/Documents/CSS-5130/se
022-04-10-071106-339-1' because it contains no reports.
scan-build: No bugs found.
namruth:securecoding$ ./a.out
Original First Number is : 10
 Original Second Number is : 20
 Swapped First Number is : 20
Swapped Second Number is : 10
namruth:securecoding$
```

**Changes made to the code**

- Initializing ptr_to_first_number and ptr_to_second_number to addresses of first_number and second_number. Line number 36 and 37
- Checking for ptr1 and ptr2 in the swap function before accessing them. Line number 13.
- Implementing the swap logic from line number 22 to 24. Logic used – If a and b are two numbers that needs to be swapped the, a =a+b, b = a-b, a = a-b. a and b will be swapped without using a third variable.

# CodeN1116.c

**Output of clang static analyzer report**

### securecoding - scan-build results

| | |
|---|---|
| **User:** | namruth@DESKTOP-LETJM2M |
| **Working Directory:** | /mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_project5/securecoding |
| **Command Line:** | gcc codeN1116.c |
| **Clang Version:** | clang version 10.0.0-4ubuntu1 |
| **Date:** | Sun Apr 10 07:19:13 2022 |

**Bug Summary**

| Bug Type | Quantity | Display? |
|---|---|---|
| All Bugs | 2 | ☑ |
| **Logic error** | | |
| Dereference of null pointer | 1 | ☑ |
| **Unix API** | | |
| Allocator sizeof operand mismatch | 1 | ☑ |

**Reports**

| Bug Group | Bug Type ▾ | File | Function/Method | Line | Path Length | |
|---|---|---|---|---|---|---|
| Unix API | Allocator sizeof operand mismatch | codeN1116.c | main | 57 | 1 | View Report |
| Logic error | Dereference of null pointer | codeN1116.c | even_number | 14 | 4 | View Report |

**Mapping the bug to CWE known weaknesses and how to fix it**

This bug can be attributed CWE-476: NULL Pointer Dereference. More details about it can be found here - https://cwe.mitre.org/data/definitions/476.html

It can also be attributed to CWE-467: Use of sizeof() on a Pointer Type. More details about it can be found here - https://cwe.mitre.org/data/definitions/467.html

This weakness can be fixed by having proper initialization and checking for pointers if they are NULL before dereferencing them and using the right size of memory allocation inside the malloc function.

**Fixing the clang errors and output of the program obtained**

**This is the fixed code:**

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

// custom method to check for even or odd number
void even_number(int *ptr1, char *ptr2) {
    if (!ptr1 || !ptr2){
      printf("One of the pointers passed is NULL\n");
      return;
    }

    printf("Original First Number is : %d \n ", (*ptr1));
    printf("Choice is : %c \n ", (*ptr2));

    if (*ptr1 % 2 == 0)//add logic for checking even number inside if condition ()
    {
        printf("Numbner %d is Even : \n", (*ptr1));
        printf("Confirmed choice is : %c \n", (*ptr2));
    } else {
        printf("Odd Number\n");
    }

}

//another custom array printing an array
void array_print(int *ptr3, int n) {
    if (!ptr3){
      printf("One of the pointers passed is NULL\n");
      return;
    }
    for (int i = 0; i < n + 1; i++) {
        ptr3[i] = i;
        printf("Array is:%d\n ", ptr3[i]);
    }
}

int main() {
    // variable initializations (Hint: Missing values?????)
    int first_number = 17;
    char choice = 'Y';
    int *ptr_to_arr;
    int size_array = 10;

    int *ptr_to_first_number = &first_number;
    char *ptr_to_char = &choice;
    ptr_to_arr = (int *) malloc(size_array * sizeof(int));
    // Hint:: add and assign the references to the pointers here:
```

```c

int main() {
    // variable initializations (Hint: Missing values?????)
    int first_number = 17;
    char choice = 'Y';
    int *ptr_to_arr;
    int size_array = 10;

    int *ptr_to_first_number = &first_number;
    char *ptr_to_char = &choice;
    ptr_to_arr = (int *) malloc(size_array * sizeof(int));
    even_number(ptr_to_first_number, ptr_to_char);
    array_print(ptr_to_arr, size_array);

    free(ptr_to_arr);
    return 0;
}

```

**And the output obtained:**

```
namruth:securecoding$ scan-build -o . gcc codeN1116.c
scan-build: Using '/usr/lib/llvm-10/bin/clang' for static analysis
scan-build: Removing directory '/mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_
022-04-10-072647-374-1' because it contains no reports.
scan-build: No bugs found.
namruth:securecoding$ ./a.out
Original First Number is : 17
 Choice is : Y
 Odd Number
Array is:0
 Array is:1
 Array is:2
 Array is:3
 Array is:4
 Array is:5
 Array is:6
 Array is:7
 Array is:8
 Array is:9
 Array is:10
 namruth:securecoding$
```

**Changes made to the code**

- Including stdlib.h as it was missing, and it is needed functions involving memory allocation and others. Line number 10
- Checking if ptr1 or ptr2 is NULL before accessing them. Line number 14.
- Adding the logic for checking even number. Line number 22. Even numbers are divisible by 2.
- Checking if ptr3 is NULL before accessing it. Line number 34
- Initializing first_number, size_array, ptr_to_first_number, ptr_to_char to their respective values. Line number 46,49,51,52
- Fixing the malloc function to have proper size of memory allocation. Line number 53.
- Freeing the memory allocated using malloc. Line number 57

# CodeN1118.c

**Output of clang static analyzer report**

## securecoding - scan-build results

| | |
|---|---|
| **User:** | namruth@DESKTOP-LETJM2M |
| **Working Directory:** | /mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_project5/securecoding |
| **Command Line:** | gcc codeN1118.c |
| **Clang Version:** | clang version 10.0.0-4ubuntu1 |
| **Date:** | Sun Apr 10 07:35:28 2022 |

### Bug Summary

| Bug Type | Quantity | Display? |
|---|---|---|
| **All Bugs** | 3 | ☑ |
| **Logic error** | | |
| Array subscript is undefined | 1 | ☑ |
| Uninitialized argument value | 1 | ☑ |
| **Unix API** | | |
| Allocator sizeof operand mismatch | 1 | ☑ |

### Reports

| Bug Group | Bug Type ▾ | File | Function/Method | Line | Path Length | |
|---|---|---|---|---|---|---|
| Unix API | Allocator sizeof operand mismatch | codeN1118.c | main | 32 | 1 | View Report |
| Logic error | Array subscript is undefined | codeN1118.c | array_print | 15 | 3 | View Report |
| Logic error | Uninitialized argument value | codeN1118.c | main | 37 | 2 | View Report |

**Mapping the bug to CWE known weaknesses and how to fix it**

This bug can be attributed to CWE-457: Use of Uninitialized Variable. More details about it can be found here - https://cwe.mitre.org/data/definitions/457.html

It can also be attributed to CWE-467: Use of sizeof() on a Pointer Type. More details about it can be found here - https://cwe.mitre.org/data/definitions/467.html

This weakness can be fixed by having the right initialization value and then checking the values before accessing or printing them and using the right size of memory allocation inside the malloc function.

**Fixing the clang errors and output of the program obtained**

**This is the fixed code:**

```c
//MACROS
#include<stdio.h>
#include<stdlib.h>

//another custom array printing an array
void array_print(int* ptr3, int n)
{
    int sub = 0;
    if (!ptr3){
      printf("One of the pointers passed is NULL\n");
      return;
    }
    for(int i=0; i<n; i++)
    {
        ptr3[sub] = i;
        sub=sub+1;
        printf("Array is:%d\n ",ptr3[i]);
    }
}

int main(){
    // variable initializations
    int* ptr_to_arr;
    int size_array = 10;

    //Note for developers(students): use variables in method using defined pointers (not directly)

    ptr_to_arr = (int*)malloc(sizeof(int)*size_array);
    // Hint:: add and assign the references to the pointers here:

    //function call statements
    array_print(ptr_to_arr,size_array);
    free(ptr_to_arr);
    return 0;


}
```

**And the output obtained:**

```
 namruth:securecoding$ scan-build -o . gcc codeN1118.c
scan-build: Using '/usr/lib/llvm-10/bin/clang' for static analysis
scan-build: Removing directory '/mnt/c/Users/namru/OneDrive/Documents/CSS-5130/se
022-04-10-074252-401-1' because it contains no reports.
scan-build: No bugs found.
namruth:securecoding$ ./a.out
Array is:0
 Array is:1
 Array is:2
 Array is:3
 Array is:4
 Array is:5
 Array is:6
 Array is:7
 Array is:8
 Array is:9
 namruth:securecoding$
```

**Changes made to the code**

- Including stdlib.h as it was missing, and it is needed functions involving memory allocation and others. Line number 10
- Checking if ptr3 is NULL before accessing it. Line number 13
- Initializing sub to 0 as it is used in the for loop and to avoid garbage values.
- Changing for loop value to n to avoid buffer overflow. Line number 17.
- Initializing size_array to 10. Line number 28
- Fixing the malloc function to have proper size of memory allocation. Line number 32.
- Freeing the memory allocated using malloc. Line number 36.

# CodeN1120.c

**Output of clang static analyzer report**

## securecoding - scan-build results

| | |
|---|---|
| **User:** | namruth@DESKTOP-LETJM2M |
| **Working Directory:** | /mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_project5/securecoding |
| **Command Line:** | gcc codeN1120.c |
| **Clang Version:** | clang version 10.0.0-4ubuntu1 |
| **Date:** | Sun Apr 10 07:47:54 2022 |

### Bug Summary

| Bug Type | Quantity | Display? |
|---|---|---|
| **All Bugs** | **1** | ☑ |
| **Logic error** | | |
| Dereference of null pointer | 1 | ☑ |

### Reports

| Bug Group | Bug Type ▾ | File | Function/Method | Line | Path Length | |
|---|---|---|---|---|---|---|
| Logic error | Dereference of null pointer | codeN1120.c | reverse_number | 13 | 4 | View Report |

**Mapping the bug to CWE known weaknesses and how to fix it**

This bug can be attributed CWE-476: NULL Pointer Dereference. More details about it can be found here - https://cwe.mitre.org/data/definitions/476.html

This weakness can be fixed by having proper initialization and checking for pointers if they are NULL before dereferencing them.

**Fixing the clang errors and output of the program obtained**

**This is the fixed code:**

```c
// custom method to reverse a number (without using third variable/pointer) passed to it as argu
void reverse_number(int *ptr1, char* ptr2){
    if (!ptr1 || !ptr2){
      printf("One of the pointers passed is NULL\n");
      return;
    }
    int reverse = 0; //will be used as reversing variable.
    int remainder = 0;
    printf("Original First Number is : %d \n ", (*ptr1));
    printf("Choice is : %c \n ", (*ptr2));

    if(*ptr2 == 'Y')
    {
        while(*ptr1 > 0)
        {
          remainder = *ptr1%10;
          reverse = reverse*10 + remainder;
          *ptr1 /= 10;
        }
        *ptr1=reverse;
        //print statements
        printf("Reversed First Number is : %d \n", (*ptr1));
        printf("Confirmed choice is : %c \n", (*ptr2));
    }
    else{
        printf("Sorry Not allowed\n");
    }

}

int main(){
    // variable initializations
    int first_number = 12345;
    char choice = 'Y';

    int* ptr_to_first_number = &first_number;
    char *ptr_to_char = &choice;
    // Hint:: add and assign the references to the pointers here:

```

**And the output obtained:**

```
 namruth:securecoding$ scan-build -o . gcc codeN1120.c
scan-build: Using '/usr/lib/llvm-10/bin/clang' for static analysis
scan-build: Removing directory '/mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_codin
022-04-10-075038-428-1' because it contains no reports.
scan-build: No bugs found.
namruth:securecoding$ ./a.out
Original First Number is : 12345
 Choice is : Y
 Reversed First Number is : 54321
Confirmed choice is : Y
```

**Changes made to the code**

- Checking if ptr1 or ptr2 is NULL before accessing them. Line number 12.
- Adding reverse the number logic. Line number 25 to 27. Logic is to divide the number by 10 and the remainder obtained will be the last digit in the number. Then storing the number into the reverse variable.
- Initializing first_number to 12345.
- Initializing ptr_to_first_number and ptr_to_char to their respective addresses.

# CodeN1122.c

**Output of clang static analyzer report**

## securecoding - scan-build results

| | |
|---|---|
| User: | namruth@DESKTOP-LETJM2M |
| Working Directory: | /mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_project5/securecoding |
| Command Line: | gcc codeN1122.c |
| Clang Version: | clang version 10.0.0-4ubuntu1 |
| Date: | Sun Apr 10 07:55:22 2022 |

### Bug Summary

| Bug Type | Quantity | Display? |
|---|---|---|
| All Bugs | 1 | ☑ |
| Memory error | | |
| Memory leak | 1 | ☑ |

### Reports

| Bug Group | Bug Type | File | Function/Method | Line | Path Length | |
|---|---|---|---|---|---|---|
| Memory error | Memory leak | codeN1122.c | buildMatrix | 21 | 13 | View Report |

**Mapping the bug to CWE known weaknesses and how to fix it**

This bug can be attributed CWE-401: Missing Release of Memory after Effective Lifetime. More details about it can be found here - https://cwe.mitre.org/data/definitions/401.html

This weakness can be fixed by freeing the memory once you are done using it to avoid memory leaks

**Fixing the clang errors and output of the program obtained**

**This is the fixed code:**

```c
#define MAX_LEN 10

void buildMatrix(unsigned int matrix_size) {
    char **square = calloc(matrix_size, sizeof(char *));
    for (int i = 0; i < matrix_size; ++i) {
        square[i] = calloc(matrix_size, sizeof(char));
    }
    system("clear");
    printf("\tMatrix\n");

    for (int i = 0; i < matrix_size; i++) {
        printf("\n");
        for (int j = 0; j < matrix_size; j++) {
            square[i][j] = i + j;
            printf("  %d  |", i + j);
        }
    }
    printf("\n");
    free(square);
}

int main() {
    int matrix_size = 3;
    printf("Enter size of n x n matrix");
    scanf("%d", &matrix_size);
    if (matrix_size > MAX_LEN) {
        printf("Exceeded max length!");
    }
    buildMatrix(matrix_size);
    return 0;

}
```

**And the output obtained:**

```
       Matrix

  0  |  1  |  2  |
  1  |  2  |  3  |
  2  |  3  |  4  |
namruth:securecoding$ scan-build -o . gcc codeN1122.c
scan-build: Using '/usr/lib/llvm-10/bin/clang' for static analysis
scan-build: Removing directory '/mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_cod
022-04-10-075850-472-1' because it contains no reports.
scan-build: No bugs found.
namruth:securecoding$
```

**Changes made to the code**

- Using “clear” in system call as the program is being run on a Linux machine. Line number 11
- Freeing the square memory once the matrix is printed to avoid memory leaks. Line number 22.

# CodeN1124.c

**Output of clang static analyzer report**

**securecoding - scan-build results**

| | |
|---|---|
| User: | namruth@DESKTOP-LETJM2M |
| Working Directory: | /mnt/c/Users/namru/OneDrive/Documents/CSS-5130/secure_coding_project5/securecoding |
| Command Line: | gcc codeN1124.c |
| Clang Version: | clang version 10.0.0-4ubuntu1 |
| Date: | Sun Apr 10 08:01:19 2022 |

**Bug Summary**

| Bug Type | Quantity | Display? |
|---|---|---|
| All Bugs | 1 | ☑ |
| Logic error | | |
| Branch condition evaluates to a garbage value | 1 | ☑ |

**Reports**

| Bug Group | Bug Type | File | Function/Method | Line | Path Length | |
|---|---|---|---|---|---|---|
| Logic error | Branch condition evaluates to a garbage value | codeN1124.c | main | 10 | 2 | View Report |

**Mapping the bug to CWE known weaknesses and how to fix it**

This bug can be attributed to CWE-457: Use of Uninitialized Variable. More details about it can be found here - https://cwe.mitre.org/data/definitions/457.html

This weakness can be fixed by having the right initialization value and then checking the values before accessing or printing them.

**Fixing the clang errors and output of the program obtained**

**This is the fixed code:**

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

int main(int argc, char **argv) {

    int *px = (int*)malloc(sizeof(argc));
    float foo;

    if (px) {
        foo = 3.5;
        *px = argc - 1;
        if (*px == 1) {
            printf("%6.1f", foo);
        } else {
            printf("%6.1f", 100.00 / *px);
        }
        free(px);
    }
    return 0;
}
```

**And the output obtained:**

```
  50.0namruth:securecoding$ scan-build -o . gcc codeN1124.c
scan-build: Using '/usr/lib/llvm-10/bin/clang' for static analysis
scan-build: Removing directory '/mnt/c/Users/namru/OneDrive/Documents/CSS-5
022-04-10-081244-544-1' because it contains no reports.
scan-build: No bugs found.
namruth:securecoding$ ./a.out 2 3
  50.0namruth:securecoding$ ./a.out 2
   3.5namruth:securecoding$ ./a.out
   infnamruth:securecoding$
```

**Changes made to the code**

- Added line number 7 which will dynamically allocate memory to px with the size of argc which is an int type.

## GitHub repository link

https://github.com/namruth/project5_secure_coding_private