Homework – 4 OPTIONAL

Problem 1

1. **What the program wants to achieve**? – Foo function will take a string from a character array and will copy the string to "buffer[12]" with the help of "strcpy".
2. **Bug in the program** – We know that "strcpy" function does not do boundary checking while copying. Thus, it is a vulnerability which can cause buffer overflow.

main.c

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

void foo(char* str)
{
  char buffer[12];
  strcpy(buffer, str);
}

int main(int argc, char*argv[])
{
  foo(argv[5]);
  printf("returned properly\n\n");
  return 0;
}
```

```
*** stack smashing detected ***: ./a.out terminated
Aborted (core dumped)


...Program finished with exit code 134
Press ENTER to exit console.
```

3. Analyzing the code by the help of gdb - we find the following things:
   - On printing the values of char*argv[]
   - argv[5] = "`GOROOT=/usr/local/go`"
   - converting this string to unsigned integer type - \717\982\797\984\614\711\711\511\447\108\111\999\710\847\103\111 – 48 bits – need 16 places to be fully copied
   - buffer[12] = "`\000\000\000\000\340\353\377\377\377\177\000\000`" – 36 bits – 12 places

- Thus while copying int(char argv[5]) to buffer[12], stack smashing is detected.

```
(gdb) p *argv@20
$1 = {0x7fffffffeeb8 "/home/a.out", 0x0, 0x7fffffffeec4 "LANGUAGE=en_US:en",
  0x7fffffffeed6 "HOSTNAME=Check", 0x7fffffffeee5 "HOME=/",
  0x7fffffffeeec "GOROOT=/usr/local/go", 0x7fffffffef01 "TERM=xterm",
  0x7fffffffef0c "COLUMNS=80",
  0x7fffffffef17 "PATH=/opt/swift/swift-5.0-RELEASE-ubuntu14.04/usr/bin/:/usr
/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
  0x7fffffffef8b "DISPLAY=:1", 0x7fffffffef96 "LANG=en_US.UTF-8",
  0x7fffffffefa7 "DEBIAN_FRONTEND=noninteractive",
  0x7fffffffefc6 "LC_ALL=en_US.UTF-8", 0x7fffffffefd9 "PWD=/home",
  0x7fffffffefe3 "LINES=24", 0x0,
  0x21 <error: Cannot access memory at address 0x21>,
  0x7ffff7ffa000 "\177ELF\002\001\001",
  0x10 <error: Cannot access memory at address 0x10>,
  0x1f8bfbff <error: Cannot access memory at address 0x1f8bfbff>}
```

4. **Solution** – If buffer[16], there will be no overflow and string will be copied.

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

void foo(char* str)
{
  char buffer[16];
  strcpy(buffer, str);
}

int main(int argc, char*argv[])
{
  foo(argv[5]);
  printf("returned properly\n\n");
  return 0;
}
```

```
returned properly



...Program finished with exit code 0
Press ENTER to exit console.
```

Problem 2

1. **What program wants to achieve** – to read the string at argv[1], measure the length of the string using “strlen” and allocate memory in the heap using “malloc”, using it as an “name” array.
2. **Bug** – The memory location of argv[1] i.e, 0x0 is inaccessible if argument passed is 1. Thus strlen trying to access inaccessible memory is causing segmentation fault.

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

int main(int argc, char* argv[])
{
  char* name = malloc(strlen(argv[1])) ;
  name = strcpy(name,argv[1]);
  printf("%s\n", name);
  return EXIT_SUCCESS;
}
```

input | Debug Console

start | pause | continue | step over | step into | step out | help

```
(gdb) p *argc@12
Cannot access memory at address 0x1
(gdb) p *argv@12
$1 = {0x7fffffffeeb8 "/home/a.out", 0x0, 0x7fffffffeec4 "LANGUAGE=en_US:en",
  0x7fffffffeed6 "HOSTNAME=Check", 0x7fffffffeee5 "HOME=/",
  0x7fffffffeeec "GOROOT=/usr/local/go", 0x7fffffffef01 "TERM=xterm",
  0x7fffffffef0c "COLUMNS=80",
  0x7fffffffef17 "PATH=/opt/swift/swift-5.0-RELEASE-ubuntu14.04/usr/bin/:/usr
/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
  0x7fffffffef8b "DISPLAY=:1", 0x7fffffffef96 "LANG=en_US.UTF-8",
  0x7fffffffefa7 "DEBIAN_FRONTEND=noninteractive"}
(gdb) p *argv[1]
Cannot access memory at address 0x0
(gdb)
```

3. **Solution** - create two conditions to do the same malloc application for when argc = 1 and when argc >= 1.

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

int main(int argc, char* argv[])
{
  if(argc>1)
  {
  char* name = malloc(strlen(argv[1] + 1));
  name = strcpy(name,argv[1]);
  printf("%s\n", name);
  }
  else
  {char* name = malloc(strlen(argv[2] + 1));
  name = strcpy(name,argv[2]);
  printf("%s\n", name);}
  return EXIT_SUCCESS;
}
```

- **When argc = 1** | **When argc > 1**

```
start  pause  continue  step over  step into  step out
Reading symbols from a.out...done.
/usr/share/gdb/gdbinit: No such file or directory.
(gdb) run
Starting program: /home/a.out

Breakpoint 1, main (argc=1, argv=0x7fffffffecd8) at main.c:7
7         if(argc>1)
```

```
start  pause  continue  step over  step into  step out  help
Breakpoint 1, main (argc=1, argv=0x7fffffffecd8) at main.c:7
7         if(argc>1)
(gdb) c
Continuing.
LANGUAGE=en_US:en
[Inferior 1 (process 2927) exited normally]
(gdb)
```

```
Reading symbols from a.out...done.
/usr/share/gdb/gdbinit: No such file or directory.
(gdb) run Hello This is IST:543
Starting program: /home/a.out Hello This is IST:543

Breakpoint 1, main (argc=5, argv=0x7fffffffeca8) at main.c:7
7         if(argc>1)
```

```
Breakpoint 1, main (argc=5, argv=0x7fffffffeca8) at main.c:7
7         if(argc>1)
(gdb) c
Continuing.
Hello
[Inferior 1 (process 5824) exited normally]
(gdb)
```

Problem 3

1. **What the program wants to achieve** – define a function which will be fed an array, length of array whenever called in the main function and it will give back the highest number from the array.
2. **Issue with the program** – The function is returning the first element from the array and not the highest number.

Run | Debug | Stop | Share | Save | { } Beautify

main.c

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

int findandreturnMaxVal(int *a1, int len, int max) {
  int i;
  if (!a1 || (len <=0) ) {
    return -1;
  }

  max = a1[0];
  for(i=1; i <= len; i++) {
    if (max < a1[i]) {
      max = a1[i];
    }
  }
  return 0;
}

int main(int argc, char *argv[]) {
  int arr[5] = { 11, 25, 47, 8, 68 };
  int max = arr[0];

  if (findandreturnMaxVal(arr, 5, max) != 0 ) {
    printf("unknown error\n");
    exit(1);
  }

  printf("The max value in the array is %d\n", max);
  return 0;
}
```

```
The max value in the array is 11


...Program finished with exit code 0
Press ENTER to exit console.
```

3. **Bug** – on analyzing the code, following irregularities were noticed.
    - findandreturnMaxVal has a “max” argument already fed into it.
    - In the for loop, i increments till len, but value of element of corresponding value of i in the array is garbage value after a1[len – 1]. So max stores a garbage value in the last loop.
    - findandreturnMaxVal Function does not return max value, it only calculates the max value.
4. **Solution** –
    - Remove the “max” argument from the findandreturnMaxVal function.
    - Change the return value from return 0 to return max.
    - Change the for loop from (i=1; i <= len; i++) to (i=1; i < len; i++).
    - In the main function, remove the max argument when findandreturnMaxVal function is called.
    - In printf, change the max to findandreturnMaxVal(arr, 5).
    - The program then returns the largest value of the array.

```c
  max = a1[0];
  for(i=1; i < len; i++) {
    if (max < a1[i]) {
      max = a1[i];
    }
  }

  return max;
}

int main(int argc, char *argv[]) {
  int arr[5] = { 11, 25, 47, 8, 68 };

  if (findandreturnMaxVal(arr, 5) == 0 ) {
    printf("unknown error\n");
    exit(1);
  }

  printf("The max value in the array is %d\n", findandreturnMaxVal(arr, 5));
  return 0;
}
```

input

```
The max value in the array is 68


...Program finished with exit code 0
Press ENTER to exit console.
```