40183863
Ricardo Sánchez Marchand

Assessed Practical 1

ELE8094 SwA

November 2016

rsanchezmarchand01@qub.ac.uk

Q1

1 What is the value of result in the code
Result float :1.605703

2 Explain the flaw
When an int is divided the result is an int.
In this case when a/b is done the result is 0, so when is multiplied for c the result is still 0.

with a int
result = (23/45)*3.1415
result= (0)*3.1415
result=0

with a float
result = (23/45)*3.1415
result= (.5111)*3.1415
result= **1.605703**

3 Define a variable fix and write a statement with the correct expression

```
int main(void)
{
  float a = 23;
  short b = 45;
  float c = 3.1415927;
  float result;

  result = a / b * c;
  printf("%f",result);
  return 0;
```

Q2

What is the value of result in the code
Before the fix 0
After the fix 1

Explain the flaw
Conversions can occur implicitly as required by an operation. Conversions can lead to lost or misinterpreted data. In this case the unsigned int b is converted to an integer for the comparison but that generates misinterpreted data. [1]

Define a variable fix and write a statement with the correct expression

```
int main(void)
{
  int a = -1;
  int b = 1;

  printf("%d\n", a < b);

  return 0;
}
```

[1] https://www.securecoding.cert.org/confluence/display/c/INT02-C.+Understand+integer+conversion+rules

Q3

Explain the vulnerability in the code below.
The vulnerability is called buffer overflow, this occur when the size of the memory is not enough for the data that is being copied. If the size is fixed, 10 for example, if is not enough it would override other locations of the memory. This can be used by an attacker to change the functionality of the system.

Write code to provide a fix.

```
int main(void)
{
  char *pFirstName = "FirstName ";
  char *pLastName  = "LastName";
  char name[strlen(pLastName)+strlen(pFirstName)];

  strcat(strcpy(name, pFirstName), pLastName);
  printf("Name: %s\n", name);

  return 0;
}
```

Q4

//Explanation:
//I checked char by char if their were letter or number, if so a counter increases, if we have 8 //valid characters the password is approved.

Write a function to sanitise the input

```
void getPassword(void)
{
  char password[9];
  int c=0;
  fputs("Enter Password of 8 Characters Containing Only Letters and Numbers\n", stdout);
   fgets(password, 9, stdin);
   for (int i = 0;i<8;i++){
    if (isalpha(password[i]) || isdigit(password[i])){c++;}}
    if (c!=8){
        c=0;
        printf("Please try again, your password does not match the criteria password:
%s\n",password);}
else{printf("The password matches the criteria, congratulations, you can continue password:
%s\n",password);}

  return;
}
```

Q5

```
 /*********************************************************/
/*    ELE8094 SwA Assessed Practical 2 Q5 2016          */
/*                                                      */
/* Identify and Explain the vulnerability in the        */
/* code below.  Provide a fix.                          */
/*********************************************************/
```

//the problem is that it does not have a null terminator
//this can cause undefined behavior if reading or worst in writing like Format
//String attack reference: http://wiki.c2.com/?NonNullTerminatedString

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

#define MAC_ADDRESS_LENGTH 36
#define LINELENGTH 56
#define SEPERATOR ':'


int GetMacAddress(char *MacAddress);
char outPutMacAddress[MAC_ADDRESS_LENGTH];

int main()
{

  if (1 == GetMacAddress(outPutMacAddress))
  {
    printf("Failed to get Mac address\n");
  }
  else
  {
    printf("%s\n", outPutMacAddress);
  }

  return 0;
}


int GetMacAddress(char *MacAddress)
{
  FILE *fp = NULL;
  char line[LINELENGTH];
  unsigned char counter = 1;
```

```
unsigned char i = 0;
char *address;

if (-1 == system("/sbin/ifconfig |grep --binary-files=text HWaddr >macAddress"))
{
  printf("SYSTEM_ERROR_GET_MAC_ADDRESS_FAILURE\n");
  return 1;
}
else
{
  fp=fopen("macAddress", "r");
  if(NULL == fp)
  {
    printf("Error reading macAddress file\n");
    return 1;
  }
  else
  {
    if (NULL == fgets(line, sizeof(line), fp))
    {
      printf("Error reading line from file - mac adddr\n");
      return 1;
    }
    else
    {
      address = strchr(line, SEPERATOR);
      if (NULL == address)
      {
        printf("Error in line format\n");
        return 1;
      }
      else
      {
        while ((address[counter] != 0) && (i < MAC_ADDRESS_LENGTH))
        {
          MacAddress[i] = address[counter];
          counter++;
          i++;
        }
      }
    }
    fclose(fp);
  }
}
if (-1 == remove("macAddress"))
{
```

```
    printf("Error removing file\n");
    return 1;
  }
  return 0;
}
```