COMP8050 – Security for Software Systems

Assignment 1 (50%)

For this assignment, we will use the C program provided below, **assignment1.c** . Compile it with the usual command:

```
gcc -g -O0 -mpreferred-stack-boundary=2    -m32      -fno-
stack-protector -z execstack  -D_FORTIFY_SOURCE=0
assignment1.c –o assignment1.o
```

Also ensure that you have disabled ASLR:

```
sudo sysctl -w kernel.randomize_va_space=0
```

If you have not already done so in the labs, you may need to install the following packages first too:

```
sudo apt-get install libc6-dev libc6-dev-i386 gcc-multilib
```

Question 1 – [30 Marks]

question1.c:

```
1.  #include <stdlib.h>
2.  #include <unistd.h>
3.  #include <stdio.h>
4.  #include <string.h>
5.
6.  void partialwin()
7.  {
8.    printf("Achieved 1/2!\n");
9.  }
10.
11. void fullwin()
12. {
13.   printf("Achieved 2/2!\n");
14. }
15.
16. void vuln()
17. {
18.   char buffer[36];
19.
20.   gets(buffer);
21.   printf("Buffer contents %s\n", buffer)
22. }
```

```
23.
24. int main(int argc, char **argv)
25. {
26.   vuln();
27. }
```

What you must do:

You must perform an attack on the program and cause it to output both “Achieved 1/2!” and “Achieved 2/2!” from the functions patrialwin() and fullwin(), in that order, in a single run of the program.. You must document your approach clearly in the following way:

1) Provide a large paragraph, or two, which gives a high-level description of the approach you intend to take in order to achieve your attack. It should be clear and concise. If you started with one approach, but swapped midway to another after realising something, describe both the initial idea and your final one here too.

2) Show step-by-step how you performed your attack. You should include screenshots of your input/output (e.g. using the Windows “snipping tool”) and provide short comments explaining **why** you did each action. For example:

   Sample Command (should be screenshotted with the output included): x\24x $esp
   Sample explanations for why:

   I used it to show the stack. **(BAD – this is what you did, not why you did it)**

   I used it to show the contents of 24 addresses on the stack, in order to identify the exact location of the buffer and calculate how much overflow was necessary to write over the saved base pointer. **(GOOD! – here the purpose of using a command to show the stack is explained!)**

3) **You must include a detailed diagram of the current state of the stack at each major step, starting from the initial state before your attack begins.** The diagrams should show all of the relevant elements on the stack (similar to those provided in the lecture notes).

4) You must show how you would change the code to **fix all the vulnerabilities** in the program. Provide a brief description of why your changes fix the issues.

Question 2 – [30 Marks]

question2.c:

```
1. #include <stdlib.h>
2. #include <unistd.h>
3. #include <stdio.h>
4. #include <string.h>
5.
6. int grade;
7.
8. void securegrading()
9. {
10.   if( grade < 40 )
11.   {
12.     printf("Usual grade attained.\n");
13.   }
14.   else if( grade < 100 )
15.   {
16.     printf("Excellent grade attained!\n");
17.   }
18.   else if( grade == 100 )
19.   {
20.     printf("Perfect grade attained!\n");
21.   }
22.   exit(1);
23.
24. }
25.
26. int main(int argc, char **argv)
27. {
28.   char input[48];
29.
30.   grade = 10;
31.
32.   gets(input);
33.   printf( "User input:" );
34.   printf( input );
35.
36. }
```

What you must do:

You must perform an attack on the program and cause it to run line 20 and output "Perfect grade attained!". As you can see, the securegrading() function is not normally called at all, so some form of redirection will be necessary. You must document your approach clearly in the following way:

1) Provide a large paragraph, or two, which gives a high-level description of the approach you intend to take in order to achieve your attack. It should be clear and concise. If you started with one approach, but swapped midway to another after realising something, describe both the initial idea and your final one here too.

2) Show step-by-step how you performed your attack. You should include screenshots of your input/output (e.g. using the Windows "snipping tool") and provide short comments explaining **why** you did each action. For example:

   Sample Command (should be screenshotted with the output included): x\24x $esp
   Sample explanations for why:

   I used it to show the stack. **(BAD – this is what you did, not why you did it)**

   I used it to show the contents of 24 addresses on the stack, in order to identify the exact location of the buffer and calculate how much overflow was necessary to write over the saved base pointer. **(GOOD! – here the purpose of using a command to show the stack is explained!)**

3) You must show how you would change the code to **fix all the vulnerabilities** in the program. Provide a brief description of why your changes fix the issues.

Question 3 – [25 Marks]

1) Provided a detailed description of Address Space Layout Randomization (ASLR), explaining what it is, as well as its effectiveness and limitations as a security solution.

   a. Evaluate whether or not ASLR is an effective defence against *Stack Buffer Overflow Attacks* (a.k.a. Stack Smashing).
   b. Evaluate whether or not ASLR is an effective defence against *Format String Attacks*.

2) Provided a detailed description of a Non-executable Stack, explaining what it is, as well as its effectiveness and limitations as a security solution.

   a. Evaluate whether or not a Non-executable Stack is an effective defence against *Stack Buffer Overflow Attacks* (a.k.a. Stack Smashing).
   b. Evaluate whether or not a Non-executable Stack is an effective defence against *Format String Attacks*.

Question 4 – [15 Marks]

1) Thoroughly describe what is meant by a Structured Exception Handler (SEH) Attack, including the risks posed for a program vulnerable to this type of attack.

2) Explain what *Structured Exception Handling Overwrite Protection (SEHOP)* is and clearly show how it protects against SEH Attacks.

Include all of the above in a single .pdf document. The name of the document **must be** your name followed by your student ID. e.g. "David Stynes R100000924.pdf". Penalties will be applied for incorrectly named submissions. Submit your pdf on Canvas in the submission facility located in "Assignments -> Assignment 1 Submission".

Due Date: Thursday 26th November 23:59 (Week 9)

Students may be interviewed to verify that their submissions were their own.