

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

CZ4062 Computer Security

Project Assignment Report

Group 10	
Name	Matriculation Number
Ang Jia Li	U1420967K
Ang Jun Hao	U1421127E
Neoh Tze Chiang	U1422630D
Rainy Sokhornn	U1423091B

Contents

Introduction	3
Problem 1	3
Brief Description	3
Case 1	3
Case 2	3
Case 3	4
Case 4	4
Case 5	5
Case 6	5
Case 7	6
Case 8	6
Case 9	6
Case 10	7
Case 11	8
Case 12	8
Case 13	8
Case 14	8
Case 15	9
Case 16	9
Case 17	10
Case 18	10
Case 19	10
Case 20	11
Case 21	11

Introduction

In this assignment, an Ubuntu installation with a user account having administrative privilege over the system is provided. The main objective is to learn the greybox fuzzing techniques and gain more knowledge of vulnerability issues for modern projects.

Problem 1

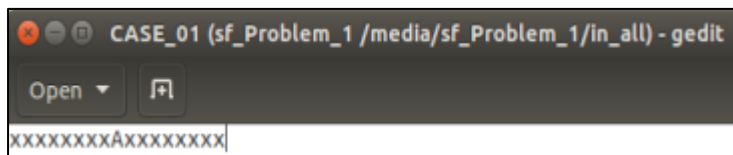
Brief Description

A simple C program (problem1.c) has been provided. It contains multiple conditions that will lead to different execution trace by seeding them with input files. There are a few different exit scenarios:

- Normal Exit (taking not more than 1s): normal execution, working with the output desired
- Timeout: Working but takes more than 1s to print the output desired.
- Crash: Not working

The fuzzing technique can help us find crashes or abnormal behaviour of which can have security implications. We shall explore the concept with the following cases with the main objective of writing proper seed inputs so that fuzzing works.

Case 1



Create a file and write “xxxxxxxxAxxxxxxxx”. The number of characters is more than 16 because of the condition “if (nread >16)” in problem1.c. The character “A” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x41” represents “A” in the ASCII table.

The execution of this input is a crash because pointer *b is pointing to NULL and it means that the pointer is not intended to point to an accessible memory location. Therefore, the next line “*b = 99” caused null point dereference hence crashing the system as shown below.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_01
CASE_01 [nread>16] crashing: null pointer dereference
Segmentation fault (core dumped)
```

Case 2



Create a file and write “xxxxxxxxxxxxxxxxxxxx”. The number of characters is more than 16 because of the condition “if (nread >16)” in problem1.c. The character “a” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x61” represents “a” in the ASCII table.

The execution of this input is a timeout exit because it takes more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_02
CASE_02 [nread>16] timeout
```

Case 3

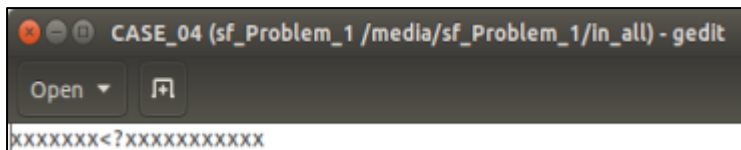


Create a file and write “xxxxxxxx*xxxxxxxxxxx”. The number of characters is more than 16 because of the condition “if (nread >16)” in problem1.c. The character “*” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x2A” represents “*” in the ASCII table.

The execution of this input is a crash because of assertion failure hence causing the abnormal program termination as shown below.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_03
CASE_03 [nread>16] assert_failure:
problem1: problem1.c:43: main: Assertion '0' failed.
Aborted (core dumped)
```

Case 4

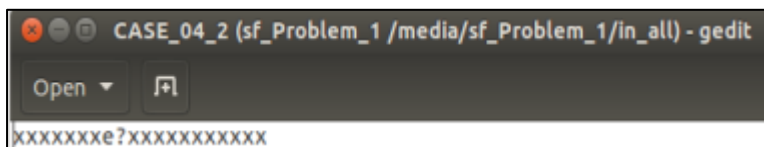


Create a file and write “xxxxxxx<?xxxxxxxxxxx”. The number of characters is more than 16 because of the condition “if (nread >16)” in problem1.c. The character “?” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x3F” represents “?” in the ASCII table. The character “<” is in the 8th position because of the condition “if (buf[7] == 0x3C)” and the fact that “0x3C” represents “<” in the ASCII table.

The execution of this input is a normal exit because it does not take more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_04
CASE_04 [nread>16] loop: k=60
```

Another Case 4



Create a file and write “xxxxxxxe?xxxxxxxxxxx”. The number of characters is more than 16 because of the condition “if (nread >16)” in problem1.c. The character “?” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x3F” represents “?” in the ASCII table. The

character “e” is in the 8th position because of the condition “else if (buf[7] > 0x64)” and the fact that “0x65” represents “e” in the ASCII table and “0x65” is greater than “0x64”.

The execution of this input is a normal exit because it does not take more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_04_2
CASE_04 [nread>16] exit 6
```

Case 5

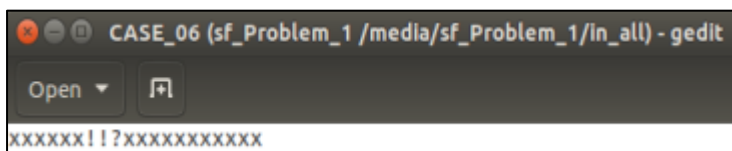


Create a file and write “xxxxxx22?xxxxxxxxxx”. The number of characters is more than 16 because of the condition “if (nread > 16)” in problem1.c. The character “?” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x3F” represents “?” in the ASCII table. The character “2” is in the 8th and 7th position because of the condition “if ((buf[6] + buf[7] == 0x64))” and the fact that “0x32” represents “2” in the ASCII table and “0x32” plus “0x32” gives “0x64”.

The execution of this input is a normal exit because it does not take more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_05
CASE_05 [nread>16] equality condition
```

Case 6

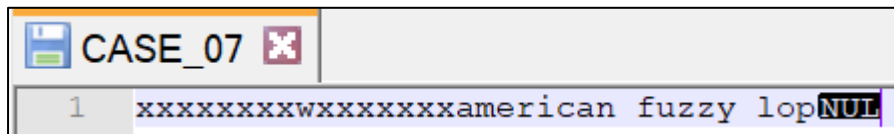


Create a file and write “xxxxxx!!?xxxxxxxxxx”. The number of characters is more than 16 because of the condition “if (nread > 16)” in problem1.c. The character “?” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x3F” represents “?” in the ASCII table. The character “!” is in the 8th and 7th position because of the else condition after “if ((buf[6] + buf[7] == 0x64))” and the fact that “0x21” represents “!” in the ASCII table and “0x21” plus “0x21” does not give “0x64”.

The execution of this input is a normal exit because it does not take more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_06
CASE_06 [nread>16] inequality condition
```

Case 7

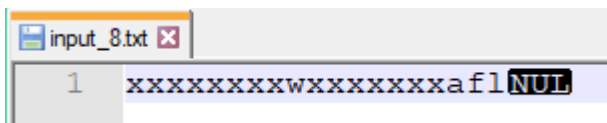


Create a file and write “xxxxxxxxxxxxxxxxamerican fuzzy lopNUL”. The NUL character is inserted via notepad++ by going to the Character Panel under Edit. The number of characters is more than 16 because of the condition “if (nread >16)” in problem1.c. The character “w” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x77” represents “w” in the ASCII table. The string “american fuzzy lop NUL” is added after 15 characters because of the string compare operator “if (strcmp(buf+16, “american fuzz lop”) == 0)”. The NUL character is important because it indicates the end of the string.

The execution of this input is a crash because of abort() function hence causing the abnormal program termination as shown below.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_07
CASE_07 Oh no, I've been caught, excellent!
Aborted (core dumped)
```

Case 8

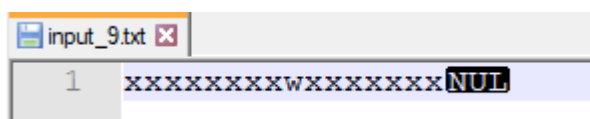


The input for case 8 is “xxxxxxxxxxxxxxxxaflNUL”. The NUL character is inserted via notepad++ by going to the Character Panel under Edit. The number of characters is more than 16 because of the condition “if (nread >16)” in problem1.c. The character “w” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x77” represents “w” in the ASCII table. The string “afl NUL” is added after 15 characters because of the string compare operator “if (strcmp(buf+16, “afl”) == 0)”. The NUL character is important because it indicates the end of the string.

The execution of this input is a crash because of abort() function hence causing the abnormal program termination as shown below.

```
user1@ntu: /media/sf_New_folder
user1@ntu:/media/sf_New_folder$ ./problem1 in_all/input_8.txt
CASE_08 oops, I surrender
Aborted
```

Case 9



```
user1@ntu: /media/sf_New_folder
user1@ntu: /media/sf_New_folder$ ./problem1 in_all/input_9.txt
CASE_09 nothing found
```

[illegible][illegible]

The number of characters is 2048 because the buffer is being increased in problem1.c. The DEL character represented by ☐ is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x7F” is “DEL” in the ASCII table.

```
user1@ntu: /media/sf_New_folder
user1@ntu:/media/sf_New_folder$ ./problem1 in_all/input_10.txt
CASE_10 can you catch me?
Floating point exception
```

Case 11

```
input_11.txt
1 xxxxxxxx☐xxxxxxxx
```

The input for case 11 is “xxxxxxxx☐xxxxxxxx”. The ☐ character represents the DEL character which is 0x7F in the ASCII table. It is in the 9th position because of the switch statement “switch(buf[8])”. The number of characters is more than 16 because of the condition “if (nread >16)” in problem1.c. However, case 11’s statement cannot be generated because in case 10, the buffer has been expanded to 2048 and it is not reversible by reading such input files. The only way to reverse it is to make changes to problem1.c. This is also perhaps a case of buffer overflow, which poses security risks as malicious hackers can use this to corrupt data, crash the program or cause the execution of malicious codes.

```
user1@ntu: /media/sf_New_folder
user1@ntu:/media/sf_New_folder$ ./problem1 in_all/input_11.txt
CASE_10 can you catch me?
Floating point exception
```

Case 12

There is no input needed for case 12 as it is printed after case 10 or case 11 which are the if-else conditions respectively. The crux in case 12 is the printing of the “floating point exception” error. The bonus value can’t be generated as “buf[8]” which is the 9th character is “DEL” and “0x7F” is “DEL” in ACSII. Therefore, “DEL” minus “DEL” is zero. And 12 / (buf[8] - 0x7F) will therefore be undefined (divide by 0 error).

Case 13

```
input_13.txt
1 xxxxxxxx☐xxxxxxxx
```

The input for case 13 is “xxxxxxxx☐xxxxxxxx”. The ☐ character represents the DEL and is in the 9th position because of the switch statement “switch(buf[8])”. But since it is a default case, any character will do, and doesn’t necessary have to be DEL. The number of characters is more than 16 because of the condition “if (nread >16)” in problem1.c.

```
user1@ntu: /media/sf_New_folder
user1@ntu:/media/sf_New_folder$ ./problem1 in_all/input_13.txt
CASE_13 [nread>16] common
```

Case 14


```
input_14.txt
1 xxxxxxxxANUL
```


The input for case 14 is "xxxxxxxANUL". The NUL character is inserted via notepad++ by going to the Character Panel under Edit. The number of characters is less than 16 because of the else condition after "if (nread >16)" in problem1.c. The character "A" is in the 9th position because of the switch statement "switch(buf[8])" and the fact that "0x41" represents "A" in the ASCII table. The NUL character indicates the end of the string.

A segmentation fault is generated due to the null pointer dereference issue. The null pointer dereference error occurred as the pointer *b with the value of Null is used (allocated a value of 99) as though it pointed to a valid memory area. It isn't the case, memory is not mapped properly and this triggered a segmentation fault error.

```
user1@ntu: /media/sf_New_folder
user1@ntu:/media/sf_New_folder$ ./problem1 in_all/input_14.txt
CASE_14 [nread<=16] crashing: null pointer dereference
Segmentation fault
```

Case 15



Create a file and write "xxxxxxxxa". The number of characters is less than 16 because of the else condition after "if (nread >16)" in problem1.c. The character "a" is in the 9th position because of the switch statement "switch(buf[8])" and the fact that "0x61" represents "a" in the ASCII table.

The execution of this input is a timeout exit because it takes more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_15
CASE_15 [nread<=16] timeout
```

Case 16



Create a file and write "xxxxxxxx*". The number of characters is less than 16 because of the else condition after "if (nread >16)" in problem1.c. The character "*" is in the 9th position because of the switch statement "switch(buf[8])" and the fact that "0x2A" represents "*" in the ASCII table.

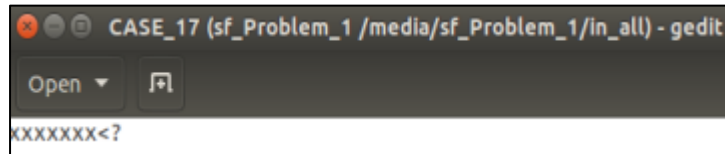
The execution of this input is a crash because of assertion failure hence causing the abnormal program termination as shown below.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_16
CASE_16 [nread<=16] assert_failure:
problem1: problem1.c:104: main: Assertion '0' failed.
Aborted (core dumped)
```

However, when the problem1.c is compiled using clang instead of gcc, the execution of the input is also a crash, but the output message is different as shown below.

```
CASE_16 [nread<=16] assert_failure:
problem1_2: problem1.c:104: int main(int, char **): Assertion `0' failed.
Aborted (core dumped)
```

Case 17

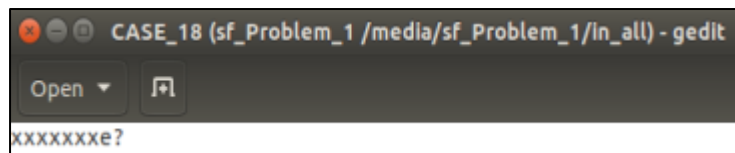


Create a file and write “xxxxxxx<?”. The number of characters is less than 16 because of the else condition after “if (nread >16)” in problem1.c. The character “?” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x3F” represents “?” in the ASCII table. The character “<” is in the 8th position because of the condition “if (buf[7] == 0x3C)” and the fact that “0x3C” represents “<” in the ASCII table.

The execution of this input is a normal exit because it does not take more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_17
CASE_17 [nread<=16] loop: k=60
```

Case 18



Create a file and write “xxxxxxxe?”. The number of characters is less than 16 because of the else condition after “if (nread >16)” in problem1.c. The character “?” is in the 9th position because of the switch statement “switch(buf[8])” and the fact that “0x3F” represents “?” in the ASCII table. The character “e” is in the 8th position because of the condition “else if (buf[7] > 0x64)” and the fact that “0x65” represents “e” in the ASCII table and “0x65” is greater than “0x64”.

The execution of this input is a normal exit because it does not take more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_18
CASE_18 [nread<=16] exit 6
```

Case 19



Create a file and write "xxxxxx22?". The number of characters is less than 16 because of the else condition after "if (nread >16)" in problem1.c. The character "?" is in the 9th position because of the switch statement "switch(buf[8])" and the fact that "0x3F" represents "?" in the ASCII table. The character "2" is in the 8th and 7th position because of the condition "if ((buf[6] + buf[7] == 0x64)" and the fact that "0x32" represents "2" in the ASCII table and "0x32" plus "0x32" gives "0x64".

The execution of this input is a normal exit because it does not take more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_19
CASE_19 [nread<=16] equality condition
```

Case 20

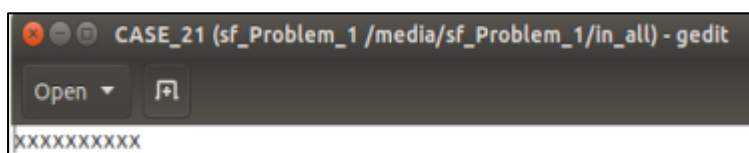


Create a file and write "xxxxxx23?". The number of characters is less than 16 because of the else condition after "if (nread >16)" in problem1.c. The character "?" is in the 9th position because of the switch statement "switch(buf[8])" and the fact that "0x3F" represents "?" in the ASCII table. The characters "2" and "3" are in the 8th and 7th position because of the else condition after "if ((buf[6] + buf[7] == 0x64)" and the fact that "0x32" represents "2" and "0x33" represents "3" in the ASCII table and "0x32" plus "0x33" does not give "0x64".

The execution of this input is a normal exit because it does not take more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_20
CASE_20 [nread<=16] inequality condition
```

Case 21



Create a file and write "xxxxxxxxxx". The number of characters is less than 16 because of the else condition after "if (nread >16)" in problem1.c. The character "x" is in the 9th position because of the default statement so any character will do.

The execution of this input is a normal exit because it does not take more than 1s and the desired output illustrated below is shown after execution.

```
user1@ntu:/media/sf_Problem_1$ ./problem1 ./in_all/CASE_21
CASE_21 [nread<16] common
```