**Updated part is in section 3 underlined and italicized. Photos are also included below it. **

1. Problem statement:
   You have been given a program called wargames.c. The code has problems with it (possibly multiple problems). Misuse of the program can result in disastrous consequences. You have been tasked with fixing the problems with the poor engineers who designed this program. You are to determine what problem(s) there are with the code, why these problems exist, and how to fix the problems.

1. Introduction

   When transitioning from high-level programming languages to low-level programming languages comes with a learning curve. Examples of high-level programing language is java, python, c#, etc. With high level programming languages, a complier/interpreter translates it to machine code but also debugs to make sure the program doesn't have errors that could causes vulnerabilities to the system. High level programming languages have automatic memory management i.e., a garbage collector which is a memory recovery feature built into these programming languages [1]. Main purpose of garbage collector is to reduce memory leaks. Memory leaks can cause performance and security issues. Low-level programming languages, such as C programming language, do not have automatic memory management which causes vulnerabilities. Memory management in C developers can allocate and deallocate memory statically and dynamically. C programming language began as a system programming language for writing operating systems. The major characteristics of C language include low-level memory access, a minimal set of keywords, and allows maximum control with minimal commands; these characteristics make C language suited for system programming such as operating system or compiler development. Since C programming allows for maximum control as a programmer it is important to be aware of these vulnerabilities that comes with this time of control such as buffer and memory vulnerabilities.

   The C programming language uses different types of memory allocation through variables, static and dynamic allocation. Static memory allocation is when a programmer declares a global or static variable. Each static or global variable defines a fixed-size block of space. The space is only ever allotted once, upon program startup as part of the exec procedure. On the other hand, dynamic allocation happens when programmer doesn't know the amount of memory or how long they will need the memory which is dependent on factor unknown at execution time. Memory allocation is important because it ensures blocks of memory are properly allocated which allows for efficient performance.

   In C programming, String data type are implemented as array of characters. Arrays have a couple disadvantages in this language such as it only allows for a fixed number of elements to be allocated with the array which is decided at the time of declaration. When declaring arrays, you must include the null character at the end of the array. In this lab report I will be analyzing the errors and solutions to these errors from the wargames program.

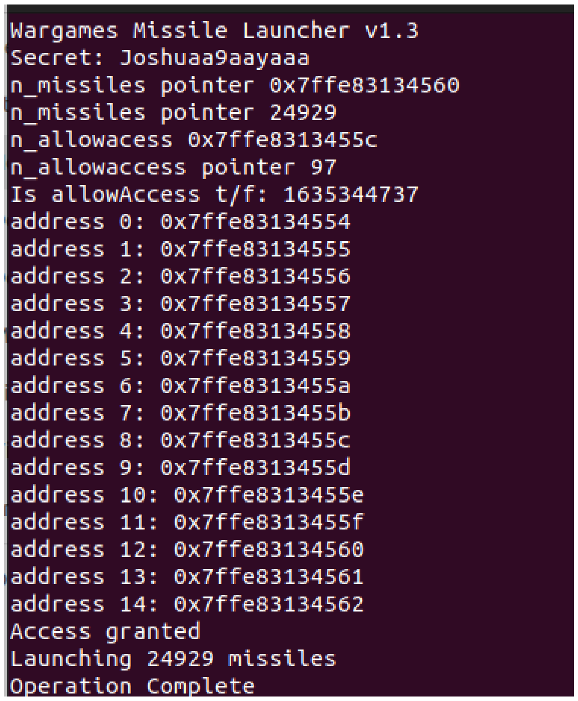
2. Errors in the Code
   a. Gets()
      i. Gets() is a known vulnerability with c programming so soon as I saw the function in the program I knew I would have to fix it. When I first ran the program, an error appeared (Fig 1.1 below). The problem with gets() is that doesn't check for

**Updated part is in section 3 underlined and italicized. Photos are also included below it. **

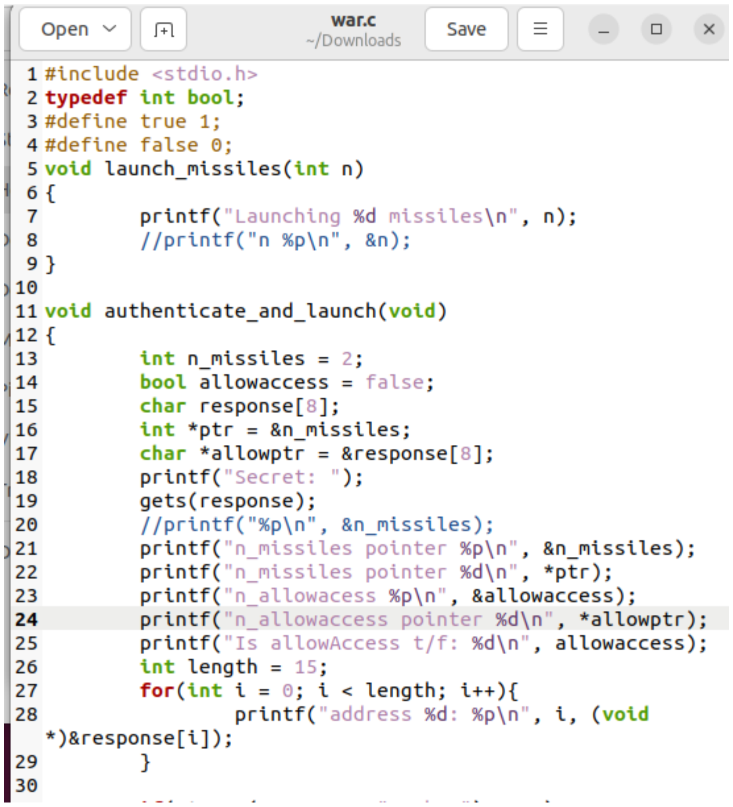
bounds of an array, ultimately causes an buffer overflow. In this case, it causes for many missiles to launch because user input is running into another part of the program. Since, the char array that the user input is allocated to only 7 characters plus a null value anything more that causes a buffer overflow. To exploit this problem, users are able to get access granted regardless of if they know the secret word, along with writing into another part of the process.

ii. <u>*For this program when the buffer overflow occurs it writes into multiple parts of the program such as n_missiles and allowaccess variables depending how far the user overwrites. The n_missiles value is overwritten if the users write over the allocated memory by four additional characters. This happens because in the process that is where the n_missiles is stored. Also, this causes for that value of the memory address to print out for the number of launching missiles. I was able to figure that out by dereferencing that variable, furthermore, whatever character that the user puts in for that memory address will be the value of the number of missiles that will be launched. Allowaccess variable is also overwritten. If the user goes over the allocated memory by one character, it causes out Boolean to change from true to false. This causes for the allowaccess to grant access rather than going into the denied access how it should. This vulnerability with the gets() causes for the user to launch missiles and have access into the wargames without having the correct secret word. These overrides cause vulnerabilities to the system because it allows the attacker to write into various other variables in memory which they should not have access to. Additionally, this causes for the system to lacks confidentiality, authentication, and Integrity of this program.*</u>

**Updated part is in section 3 underlined and italicized. Photos are also included below it. **

1. 

```
Wargames Missile Launcher v1.3
Secret: Joshuaa9aayaaa
n_missiles pointer 0x7ffe83134560
n_missiles pointer 24929
n_allowacess 0x7ffe8313455c
n_allowaccess pointer 97
Is allowAccess t/f: 1635344737
address 0: 0x7ffe83134554
address 1: 0x7ffe83134555
address 2: 0x7ffe83134556
address 3: 0x7ffe83134557
address 4: 0x7ffe83134558
address 5: 0x7ffe83134559
address 6: 0x7ffe8313455a
address 7: 0x7ffe8313455b
address 8: 0x7ffe8313455c
address 9: 0x7ffe8313455d
address 10: 0x7ffe8313455e
address 11: 0x7ffe8313455f
address 12: 0x7ffe83134560
address 13: 0x7ffe83134561
address 14: 0x7ffe83134562
Access granted
Launching 24929 missiles
Operation Complete
```

2. 

```
 1 #include <stdio.h>
 2 typedef int bool;
 3 #define true 1;
 4 #define false 0;
 5 void launch_missiles(int n)
 6 {
 7         printf("Launching %d missiles\n", n);
 8         //printf("n %p\n", &n);
 9 }
10
11 void authenticate_and_launch(void)
12 {
13         int n_missiles = 2;
14         bool allowaccess = false;
15         char response[8];
16         int *ptr = &n_missiles;
17         char *allowptr = &response[8];
18         printf("Secret: ");
19         gets(response);
20         //printf("%p\n", &n_missiles);
21         printf("n_missiles pointer %p\n", &n_missiles);
22         printf("n_missiles pointer %d\n", *ptr);
23         printf("n_allowacess %p\n", &allowaccess);
24         printf("n_allowaccess pointer %d\n", *allowptr);
25         printf("Is allowAccess t/f: %d\n", allowaccess);
26         int length = 15;
27         for(int i = 0; i < length; i++){
28                 printf("address %d: %p\n", i, (void
   *)&response[i]);
29         }
30
```

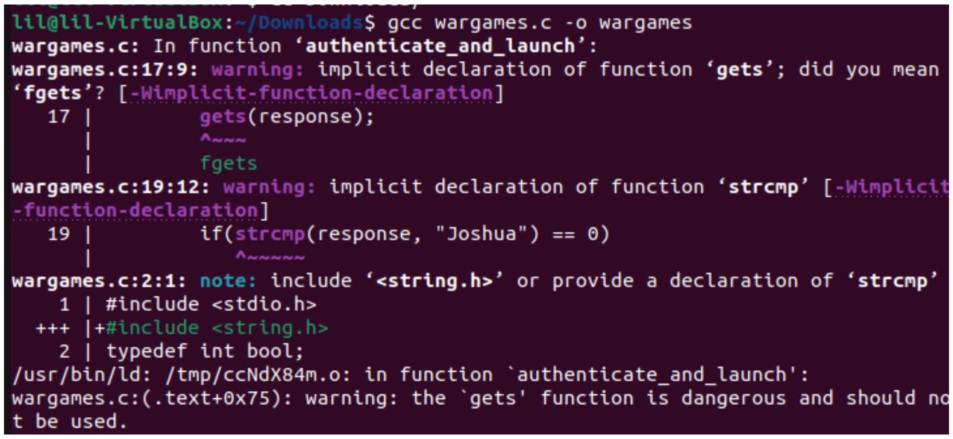
   a. This isn't the corrected code it is just how I exploited the program

iii. To solve this problem, I will replace gets() with fgets() on line 20. Fgets() allows us to avoid buffer overflows by putting limitation on end user input size. To

**Updated part is in section 3 underlined and italicized. Photos are also included below it. **

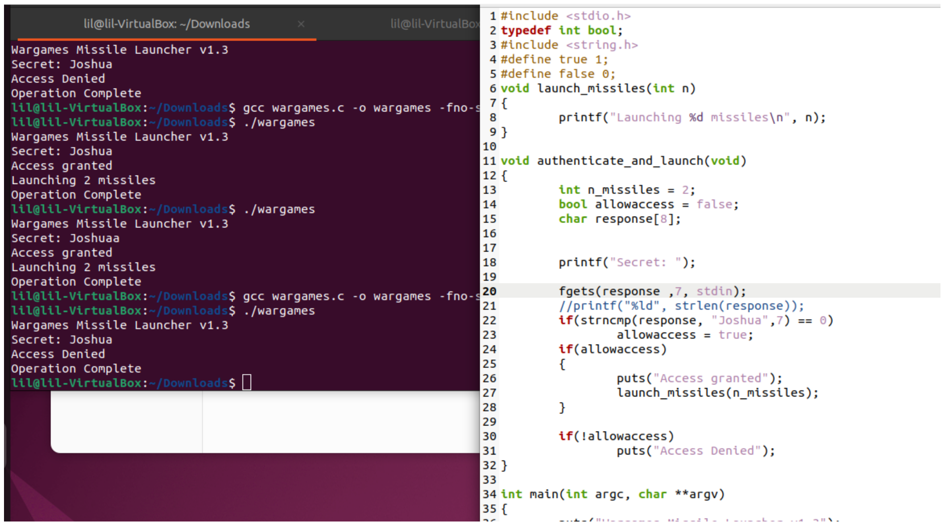
exploit this problem, I tested the program with various inputs with various characters see fig 1.3 below.

iv. When first implementing this solution I was unaware that fgets() automatically add a new line character after user input, this was causing issues with the program because it still wouldn't grant access to user if they put in "Joshua," showed in fig 1.2. This causes a problem with our next error, strcmp, in line 26.

1. 

```
lil@lil-VirtualBox:~/Downloads$ gcc wargames.c -o wargames
wargames.c: In function ‘authenticate_and_launch’:
wargames.c:17:9: warning: implicit declaration of function ‘gets’; did you mean
‘fgets’? [-Wimplicit-function-declaration]
   17 |         gets(response);
      |         ^~~~
      |         fgets
wargames.c:19:12: warning: implicit declaration of function ‘strcmp’ [-Wimplicit
-function-declaration]
   19 |         if(strcmp(response, "Joshua") == 0)
      |            ^~~~~~
wargames.c:2:1: note: include ‘<string.h>’ or provide a declaration of ‘strcmp’
    1 | #include <stdio.h>
  +++ |+#include <string.h>
    2 | typedef int bool;
/usr/bin/ld: /tmp/ccNdX84m.o: in function `authenticate_and_launch':
wargames.c:(.text+0x75): warning: the `gets' function is dangerous and should no
t be used.
```

   a. Fig 1.1

2. 

```
lil@lil-VirtualBox: ~/Downloads
Wargames Missile Launcher v1.3
Secret: Joshua
Access Denied
Operation Complete
lil@lil-VirtualBox:~/Downloads$ gcc wargames.c -o wargames -fno-s
lil@lil-VirtualBox:~/Downloads$ ./wargames
Wargames Missile Launcher v1.3
Secret: Joshua
Access granted
Launching 2 missiles
Operation Complete
lil@lil-VirtualBox:~/Downloads$ ./wargames
Wargames Missile Launcher v1.3
Secret: Joshuaa
Access granted
Launching 2 missiles
Operation Complete
lil@lil-VirtualBox:~/Downloads$ gcc wargames.c -o wargames -fno-s
lil@lil-VirtualBox:~/Downloads$ ./wargames
Wargames Missile Launcher v1.3
Secret: Joshua
Access Denied
Operation Complete
lil@lil-VirtualBox:~/Downloads$
```

```c
1 #include <stdio.h>
2 typedef int bool;
3 #include <string.h>
4 #define true 1;
5 #define false 0;
6 void launch_missiles(int n)
7 {
8         printf("Launching %d missiles\n", n);
9 }
10
11 void authenticate_and_launch(void)
12 {
13         int n_missiles = 2;
14         bool allowaccess = false;
15         char response[8];
16
17
18         printf("Secret: ");
19
20         fgets(response ,7, stdin);
21         //printf("%ld", strlen(response));
22         if(strncmp(response, "Joshua",7) == 0)
23                 allowaccess = true;
24         if(allowaccess)
25         {
26                 puts("Access granted");
27                 launch_missiles(n_missiles);
28         }
29
30         if(!allowaccess)
31                 puts("Access Denied");
32 }
33
34 int main(int argc, char **argv)
35 {
```

   a. Fig 1.2

v. Strcmp → strncmp

1. Strcmp() compares both strings until a null character of either string appears. The issue with strcmp() is that if inputs are passed that are not acceptable C-strings by accident, strcmp() keeps comparing until it reaches non-accessible memory and crashes or occasionally results in unexpected behavior. Instead, I decided to use strncmp() I can limit the search, so the search doesn't reach non-accessible memory. For this program I didn't realize this was an issue especially since we are using fgets() which makes sure the user input that we store isn't overflowing into memory. I knew this could be a potential vulnerability due to warning I received when running the program (see Fig 1.1 above). Also, I had to add #include <string.h> which allows us to manipulate array of

**Updated part is in section 3 underlined and italicized. Photos are also included below it. **

characters. To exploit this problem, I had to do excessive research to find a safe option, I was able to find one that compares two strings within a certain number of characters.

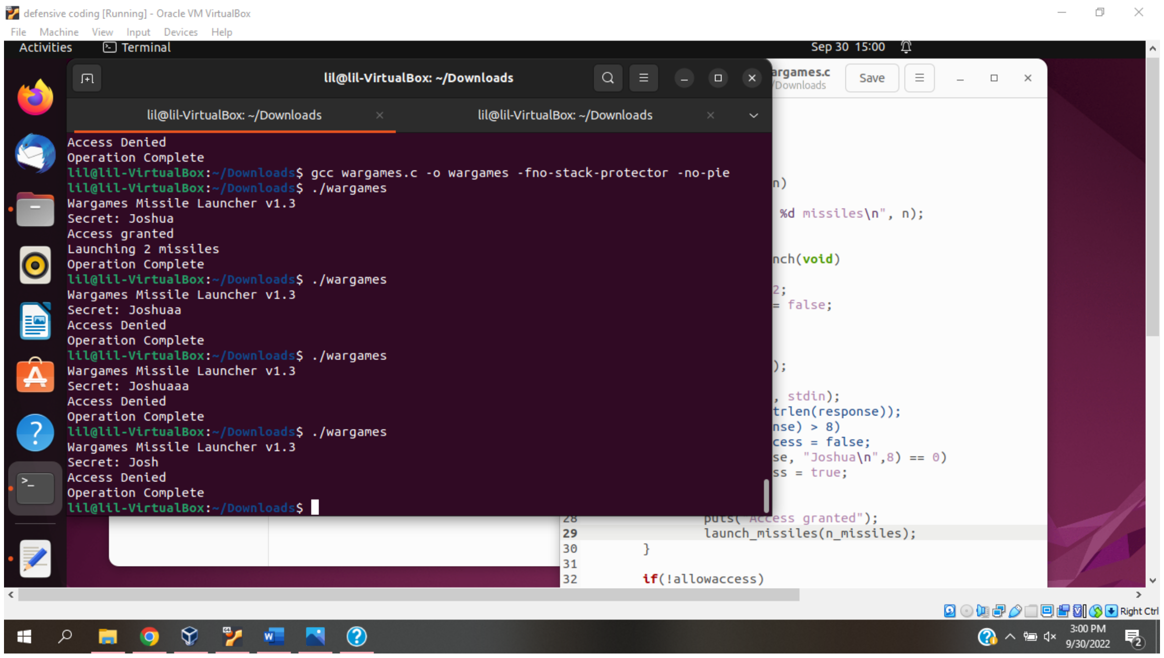
a.

vi. “Joshua\n”

1. Another error in the code on line 24, the second string “Joshua” is incorrect. This error exits because at the end of the user input when using gets has a new line character. When trying to exploit the problem with strcmp even when putting in the correct secret, “Joshua” it was still saying access denied because it was comparing “Joshua\n\0” vs “Joshua\0.” At first, I tried to compare just the first 7 characters by making changing 8 to 7 in line 26. This change cause me to run into another problem. Since it was just reading the first 7 character, if the user put in Joshua plus any characters after it access would still be granted. Due to the fact, that the two strings equal each other with the first 7 characters.

response[0]	response[1]	response[2]	response[3]	response[4]	response[5]	response[6]	response[7]
‘J’	‘o’	‘s’	‘h’	‘u’	‘a’	‘\n’	‘\0’

a. Char response array that grants access

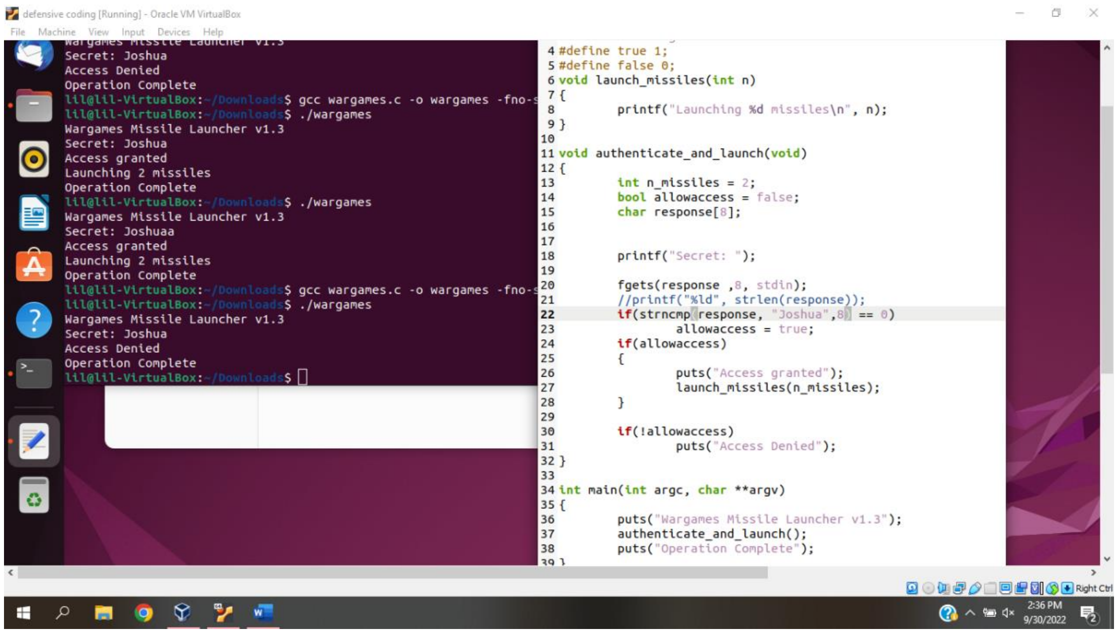

```
Access Denied
Operation Complete
lil@lil-VirtualBox:~/Downloads$ gcc wargames.c -o wargames -fno-stack-protector -no-pie
lil@lil-VirtualBox:~/Downloads$ ./wargames
Wargames Missile Launcher v1.3
Secret: Joshua
Access granted
Launching 2 missiles
Operation Complete
lil@lil-VirtualBox:~/Downloads$ ./wargames
Wargames Missile Launcher v1.3
Secret: Joshuaa
Access Denied
Operation Complete
lil@lil-VirtualBox:~/Downloads$ ./wargames
Wargames Missile Launcher v1.3
Secret: Joshuaaa
Access Denied
Operation Complete
lil@lil-VirtualBox:~/Downloads$ ./wargames
Wargames Missile Launcher v1.3
Secret: Josh
Access Denied
Operation Complete
lil@lil-VirtualBox:~/Downloads$
```

```
n)
 %d missiles\n", n);
nch(void)
2;
= false;
);
, stdin);
trlen(response));
nse) > 8)
cess = false;
se, "Joshua\n",8) == 0)
ss = true;
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if(!allowaccess)
```
defensive coding [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Sep 30 15:00
lil@lil-VirtualBox: ~/Downloads
argames.c
Downloads
Save
3:00 PM
9/30/2022

b. Testing with error fixed

**Updated part is in section 3 underlined and italicized. Photos are also included below it. **



```
Wargames Missile Launcher v1.3
Secret: Joshua
Access Denied
Operation Complete
lil@lil-VirtualBox:~/Downloads$ gcc wargames.c -o wargames -fno-s
lil@lil-VirtualBox:~/Downloads$ ./wargames
Wargames Missile Launcher v1.3
Secret: Joshua
Access granted
Launching 2 missiles
Operation Complete
lil@lil-VirtualBox:~/Downloads$ ./wargames
Wargames Missile Launcher v1.3
Secret: Joshuaa
Access granted
Launching 2 missiles
Operation Complete
lil@lil-VirtualBox:~/Downloads$ gcc wargames.c -o wargames -fno-s
lil@lil-VirtualBox:~/Downloads$ ./wargames
Wargames Missile Launcher v1.3
Secret: Joshua
Access Denied
Operation Complete
lil@lil-VirtualBox:~/Downloads$ 
```

```c
 4 #define true 1;
 5 #define false 0;
 6 void launch_missiles(int n)
 7 {
 8         printf("Launching %d missiles\n", n);
 9 }
10
11 void authenticate_and_launch(void)
12 {
13         int n_missiles = 2;
14         bool allowaccess = false;
15         char response[8];
16
17
18         printf("Secret: ");
19
20         fgets(response ,8, stdin);
21         //printf("%ld", strlen(response));
22         if(strncmp(response, "Joshua",8) == 0)
23                 allowaccess = true;
24         if(allowaccess)
25         {
26                 puts("Access granted");
27                 launch_missiles(n_missiles);
28         }
29
30         if(!allowaccess)
31                 puts("Access Denied");
32 }
33
34 int main(int argc, char **argv)
35 {
36         puts("Wargames Missile Launcher v1.3");
37         authenticate_and_launch();
38         puts("Operation Complete");
39 }
```

c. Testing with error not fixed

3. Proposed solutions

My proposed solutions have been displayed within the c program. For the first error, gets() function I decided to use fgets() instead. Before coming to this final solution I also tried to use scanf() function but I realized it has the same issue and faces the same vulnerability as gets(). Fgets() is the safeties solution because of the added parameters that only reads the number of character that is passed in. Fgets() has three parameters; char string, int n, and stream. Char string is the pointer to an array in which the user input was being stored, in our case that would be char response. Int n is the maximum number of characters to be read, also including the null character. This is partially why putting six or seven here instead of eight wouldn't work. The last parameter stream is the pointer to where the characters are read from.

For the next error with was more of a warning for our program I swapped out strcmp with strncmp instead. This adds a second check to make sure the program, especially the user's input isn't going into unallocated memory for the variable it was assigned to. Strncmp() functions has three parameters two strings and the number of characters to compare, this function returns an number which is less than, greater than or equal to zero. When it is equal to zero it means that the two strings are equal to each other. Hence why on line 26 it has == 0.

On the last error, I was able to make a quick fix to line 26 to fix this issue. I had to change "Joshua" to "Joshua\n" because I need to compare the response from fgets() which adds the newline character without changing this our return value would be a value less than zero which means that string one is less than string two. When I was exploiting errors, I changed the number of characters in the array to seven because I was only accounting for the null character but once I changed it back to eight, I was able to successfully fix the issues

**Updated part is in section 3 underlined and italicized. Photos are also included below it. **

4. Conclusion and analysis of solutions
    a. This program had a couple errors which could cause major security issues intentionally and unintentionally from the user's input. Once I was able to handle these errors, I am confident that the program will not suffer any buffer overflows. I can't be 100% worry free that an attacker will not be able to exploit my program in future, but it will be very hard to exploit this program.

Sources:

1. https://www.techtarget.com/searchstorage/definition/garbage-collection#:~:text=Garbage%20collection%20(GC)%20is%20a,longer%20needed%20by%20the%20program.
2. https://www.tutorialspoint.com/c_standard_library/c_function_fgets.htm
3. https://www.tutorialspoint.com/c_standard_library/c_function_strncmp.htm
4. https://www.geeksforgeeks.org/arrays-in-c-cpp/
5. https://www.tutorialspoint.com/difference-between-strncmp-and-strcmp-in-c-cplusplus
6. https://iq.opengenus.org/gets-vs-fgets-in-c/#:~:text=fgets()%20is%20a%20safer,e.g.%20File%20or%20standard%20input).
7. https://www.geeksforgeeks.org/scanf-and-fscanf-in-c/
8. https://www.w3schools.com/c/c_user_input.php
9. https://fresh2refresh.com/c-programming/c-strings/c-strlen-function/
10. https://www.geeksforgeeks.org/c-program-to-compare-two-strings-without-using-strcmp-function/
11. https://www.tutorialspoint.com/cprogramming/c_memory_management.htm
12. https://www.gnu.org/software/libc/manual/html_node/Memory-Allocation-and-C.html
13. https://ict.iitk.ac.in/c-the-mother-of-all-languages/#:~:text=Since%20then%2C%20C%20language%20has,also%20used%20to%20programme%20microcontrollers.
14. https://iq.opengenus.org/get-address-of-a-variable-in-c/#:~:text=In%20C%2C%20we%20can%20get,void*)%20on%20the%20address.
15. Course textbook chapter 2
16. Class PowerPoint chapter 2