**BUFFER OVERFLOW - PART 1**

After compiling the program 'examine\_stack.c', running it in the

debugger, and breaking at line 7 (in the function 'senior'), the

following is the result of multiple gdb commands to show where

we are (with line numbers of the output on the left side for

reference in the discussion that follows):

(gdb) disas senior

Dump of assembler code for function senior:

(1) 0x0804846d <+0>: push %ebp

(2) 0x0804846e <+1>: mov %esp,%ebp

(3) 0x08048470 <+3>: sub $0x10,%esp

(4) => 0x08048473 <+6>: movl $0x7530,-0x4(%ebp)

(5) 0x0804847a <+13>: mov $0xc,%eax

(6) 0x0804847f <+18>: leave

(7) 0x08048480 <+19>: ret

End of assembler dump.

(gdb) disas junior

Dump of assembler code for function junior:

(8) 0x08048481 <+0>: push %ebp

(9) 0x08048482 <+1>: mov %esp,%ebp

(10) 0x08048484 <+3>: sub $0x28,%esp

(11) 0x08048487 <+6>: mov 0xc(%ebp),%eax

(12) 0x0804848a <+9>: mov %eax,-0x1c(%ebp)

(13) 0x0804848d <+12>: mov %gs:0x14,%eax

(14) 0x08048493 <+18>: mov %eax,-0xc(%ebp)

(15) 0x08048496 <+21>: xor %eax,%eax

(16) 0x08048498 <+23>: movl $0x67756f63,-0x16(%ebp)

(17) 0x0804849f <+30>: movl $0x737261,-0x12(%ebp)

(18) 0x080484a6 <+37>: movw $0x0,-0xe(%ebp)

(19) 0x080484ac <+43>: movl $0x2,0x4(%esp)

(20) 0x080484b4 <+51>: mov 0x8(%ebp),%eax

(21) 0x080484b7 <+54>: mov %eax,(%esp)

(22) 0x080484ba <+57>: call 0x804846d <senior>

(23) 0x080484bf <+62>: mov -0xc(%ebp),%edx

(24) 0x080484c2 <+65>: xor %gs:0x14,%edx

(25) 0x080484c9 <+72>: je 0x80484d0 <junior+79>

(26) 0x080484cb <+74>: call 0x8048330 <\_\_stack\_chk\_fail@plt>

(27) 0x080484d0 <+79>: leave

(28) 0x080484d1 <+80>: ret

End of assembler dump.

(gdb) x/32xw $esp

(A) (B) (C) (D) (E)

(29) 0xbfffeff8: 0xbffff124 0xb7fffaf0 0xbffff0e0 0xb7fe79a2

(30) 0xbffff008: 0xbffff038 0x080484bf 0x000007dc 0x00000002

(31) 0xbffff018: 0x00000000 0xbffff05e 0x6f630001 0x72616775

(32) 0xbffff028: 0x00000073 0x2f37b200 0x00000000 0x00000000

(33) 0xbffff038: 0xbffff068 0x080484f0 0x000007dc 0xbffff05e

(34) 0xbffff048: 0x00000000 0xb7fc0000 0xbffff09e 0xbffff09f

(35) 0xbffff058: 0x00000001 0x00089531 0xbffff09f 0x00000000

(36) 0xbffff068: 0xbffff0b8 0x08048551 0x00000025 0x000007dc

Other various information from gdb:

(gdb) info frame

Stack level 0, frame at 0xbffff010:

eip = 0x8048473 in senior (examine\_stack.c:7); saved eip 0x80484bf

called by frame at 0xbffff040

source language c.

Arglist at 0xbffff008, args: a=2012, b=2

Locals at 0xbffff008, Previous frame's sp is 0xbffff010

Saved registers:

ebp at 0xbffff008, eip at 0xbffff00c

(gdb) print /x $esp

$5 = 0xbfffeff8

(gdb) x/4xw $esp

0xbfffeff8: 0xbffff124 0xb7fffaf0 0xbffff0e0 0xb7fe79a2

In (1) and (2), this is the standard set of instructions called

when entering a new function (in this case 'senior(int a, int b)').

Previously, in the calling function, the arguments to the senior

function as well as the return address (the address of line 17, end

of 'junior' function) were placed on the stack. (1) is pushing the old

$ebp (base pointer) onto the stack, and (2) is taking the value in

ebp and copying it in into where the stack pointer points to.

This is so $ebp will always point to the same place (the

top of the stack before we place any of 'senior''s local variables

on the stack) so that $ebp can be used to reference contents in the

stack. This way, it frees up the stack pointer for its use as a pointer

to the top of the stack, which continually changes as the stack grows

lower into memory. In (3), the function makes room for the local

variable of 'senior' by decrementing the stack pointer 16 bytes

($0x10). This what the stack looks like at line (4) (basically just

explaining like (30) in more detail):

Memory Addr Value

0xbffff014 [ 0x00000002 ] # Second argument to 'senior' (see line (30) column (E), note 0x00000002 is equivalent to 2)

0xbffff010 [ 0x000007dc ] # First argument to 'senior' (see line (30) column (D), note 0x000007dc is equivalent to 2012)

0xbffff00c [ 0x080484bf ] # Return address (see line (30) column (C), which corresponds to line (23), a part of 'junior')

0xbffff008 [ 0xbffff038 ] <--esp,ebp # Old $ebp value

After stepping once (running 'movl $0x7530,-0x4(%ebp)', moving the

literal value 30000 into the location 4 below where the base pointer

is pointing), this is what the stack looks like (the stack pointer is

still pointing to the same address):

(gdb) x/32w $esp

(A) (B) (C) (D) (E)

(37) 0xbfffeff8: 0xbffff124 0xb7fffaf0 0xbffff0e0 0x00007530

(38) 0xbffff008: 0xbffff038 0x080484bf 0x000007dc 0x00000002

(39) 0xbffff018: 0x00000000 0xbffff05e 0x6f630001 0x72616775

(40) 0xbffff028: 0x00000073 0x2f37b200 0x00000000 0x00000000

(41) 0xbffff038: 0xbffff068 0x080484f0 0x000007dc 0xbffff05e

(42) 0xbffff048: 0x00000000 0xb7fc0000 0xbffff09e 0xbffff09f

(43) 0xbffff058: 0x00000001 0x00089531 0xbffff09f 0x00000000

(44) 0xbffff068: 0xbffff0b8 0x08048551 0x00000025 0x000007dc

(gdb) disas senior

Dump of assembler code for function senior:

0x0804846d <+0>: push %ebp

0x0804846e <+1>: mov %esp,%ebp

0x08048470 <+3>: sub $0x10,%esp

0x08048473 <+6>: movl $0x7530,-0x4(%ebp)

=> 0x0804847a <+13>: mov $0xc,%eax

0x0804847f <+18>: leave

0x08048480 <+19>: ret

End of assembler dump.

Memory Addr Value

0xbffff014 [ 0x00000002 ] # Second argument to 'senior' (see line (30) column (E), note 0x00000002 is equivalent to 2)

0xbffff010 [ 0x000007dc ] # First argument to 'senior' (see line (30) column (D), note 0x000007dc is equivalent to 2012)

0xbffff00c [ 0x080484bf ] # Return address (see line (30) column (C), which corresponds to line (23), a part of 'junior')

0xbffff008 [ 0xbffff038 ] <--ebp # Old $ebp value

0xbffff004 [ 0x00007530 ] # Value of local variable 'cougars' (see line (37) column (E), note 0x7530 is equivalent to 30000)

0xbffff000 [ 0xbffff0e0 ] # Rest of space allocated to local variable

0xbfffeffc [ 0xb7fffaf0 ] #

0xbfffeff8 [ 0xbffff124 ] <--esp #

Stepping once results in the return value of the senior function

(12, which is equivalent 0xc) being placed into the $eax register.

Stepping once more calls the leave instruction, which according

to Wikipedia ('http://en.wikipedia.org/wiki/X86\_instruction\_listings')

is equivalent to 'mov %esp,%ebp' followed by 'pop %ebp'. That means

that the value pointed to by the base pointer is put into the stack pointer,

and then pops the value at the top of the stack pointed to by

the stack pointer into $ebp, thereby restoring the old $ebp value it had saved

when entering the function. Stepping once more results in the function

returning to the value found in 0xbffff00c, the address of the next instruction

after 'junior' had called 'senior'.

I will step through the rest of the program, listing the stack and disassemblies

and label the pertinent parts on the right hand side of the sections after the # marks.

At line 17 of 'examine\_stack.c'(after stepping once):

(gdb) s

junior (x=2012, y=0xbffff05e) at examine\_stack.c:17

17 }

(gdb) x/32w $esp

0xbffff010: 0x000007dc 0x00000002 0x00000000 0xbffff05e

0xbffff020: 0x6f630001 0x72616775 0x00000073 0x76b9f300

0xbffff030: 0x00000000 0x00000000 0xbffff068 0x080484f0

0xbffff040: 0x000007dc 0xbffff05e 0x00000000 0xb7fc0000

0xbffff050: 0xbffff09e 0xbffff09f 0x00000001 0x00089531

0xbffff060: 0xbffff09f 0x00000000 0xbffff0b8 0x08048551

0xbffff070: 0x00000025 0x000007dc 0x000000bf 0xb7e43883

0xbffff080: 0x0804825c 0x00000000 0xffffffff 0x65480001

(gdb) disas junior

Dump of assembler code for function junior:

0x08048481 <+0>: push %ebp

0x08048482 <+1>: mov %esp,%ebp

0x08048484 <+3>: sub $0x28,%esp # Making room for name[10] local variable

0x08048487 <+6>: mov 0xc(%ebp),%eax # Filling the variable with 'cougars'

0x0804848a <+9>: mov %eax,-0x1c(%ebp) # etc.

0x0804848d <+12>: mov %gs:0x14,%eax

0x08048493 <+18>: mov %eax,-0xc(%ebp)

0x08048496 <+21>: xor %eax,%eax

0x08048498 <+23>: movl $0x67756f63,-0x16(%ebp)

0x0804849f <+30>: movl $0x737261,-0x12(%ebp)

0x080484a6 <+37>: movw $0x0,-0xe(%ebp)

0x080484ac <+43>: movl $0x2,0x4(%esp)

0x080484b4 <+51>: mov 0x8(%ebp),%eax

0x080484b7 <+54>: mov %eax,(%esp)

0x080484ba <+57>: call 0x804846d <senior>

=> 0x080484bf <+62>: mov -0xc(%ebp),%edx

0x080484c2 <+65>: xor %gs:0x14,%edx

0x080484c9 <+72>: je 0x80484d0 <junior+79>

0x080484cb <+74>: call 0x8048330 <\_\_stack\_chk\_fail@plt>

0x080484d0 <+79>: leave

0x080484d1 <+80>: ret

End of assembler dump.

(gdb) x/1w $ebp

0xbffff038: 0xbffff068

Stepping again:

(gdb) s

sophomore (a=37, b=2012) at examine\_stack.c:24

24 }

(gdb) x/32w $esp

0xbffff040: 0x000007dc 0xbffff05e 0x00000000 0xb7fc0000

0xbffff050: 0xbffff09e 0xbffff09f 0x00000001 0x00089531

0xbffff060: 0xbffff09f 0x00000000 0xbffff0b8 0x08048551

0xbffff070: 0x00000025 0x000007dc 0x000000bf 0xb7e43883

0xbffff080: 0x0804825c 0x00000000 0xffffffff 0x65480001

0xbffff090: 0x616d616c 0x6148206e 0x00736c6c 0x00000000

0xbffff0a0: 0x00000000 0x00000000 0x00000000 0x76b9f300

0xbffff0b0: 0xb7fc03e4 0x00000001 0xbffff0e8 0x080485a1

(gdb) disas sophomore

Dump of assembler code for function sophomore:

0x080484d2 <+0>: push %ebp

0x080484d3 <+1>: mov %esp,%ebp

0x080484d5 <+3>: sub $0x28,%esp # Making room for local variable

0x080484d8 <+6>: movw $0x8,-0xa(%ebp) # Putting 8 into tiny

0x080484de <+12>: lea -0xa(%ebp),%eax

0x080484e1 <+15>: mov %eax,0x4(%esp)

0x080484e5 <+19>: mov 0xc(%ebp),%eax

0x080484e8 <+22>: mov %eax,(%esp)

0x080484eb <+25>: call 0x8048481 <junior>

=> 0x080484f0 <+30>: leave

0x080484f1 <+31>: ret

End of assembler dump.

Stepping again:

(gdb) s

freshman (a=12, b=25, c=2012) at examine\_stack.c:30

30 }

(gdb) x/32w $esp

0xbffff070: 0x00000025 0x000007dc 0x000000bf 0xb7e43883

0xbffff080: 0x0804825c 0x00000000 0xffffffff 0x65480001

0xbffff090: 0x616d616c 0x6148206e 0x00736c6c 0x00000000

0xbffff0a0: 0x00000000 0x00000000 0x00000000 0x76b9f300

0xbffff0b0: 0xb7fc03e4 0x00000001 0xbffff0e8 0x080485a1

0xbffff0c0: 0x0000000c 0x00000019 0x000007dc 0xb7e43a55

0xbffff0d0: 0x000007dc 0x0000000c 0x00000019 0xb7fc0000

0xbffff0e0: 0x080485c0 0x00000000 0x00000000 0xb7e29905

(gdb) disas freshman

Dump of assembler code for function freshman:

0x080484f2 <+0>: push %ebp

0x080484f3 <+1>: mov %esp,%ebp

0x080484f5 <+3>: sub $0x48,%esp # room for local variables

0x080484f8 <+6>: mov %gs:0x14,%eax

0x080484fe <+12>: mov %eax,-0xc(%ebp)

0x08048501 <+15>: xor %eax,%eax

0x08048503 <+17>: movl $0x616c6548,-0x2a(%ebp)

0x0804850a <+24>: movl $0x206e616d,-0x26(%ebp)

0x08048511 <+31>: movl $0x6c6c6148,-0x22(%ebp)

0x08048518 <+38>: movl $0x73,-0x1e(%ebp)

0x0804851f <+45>: movl $0x0,-0x1a(%ebp)

0x08048526 <+52>: movl $0x0,-0x16(%ebp)

0x0804852d <+59>: movl $0x0,-0x12(%ebp)

0x08048534 <+66>: movw $0x0,-0xe(%ebp)

0x0804853a <+72>: mov 0xc(%ebp),%eax

0x0804853d <+75>: mov 0x8(%ebp),%edx

0x08048540 <+78>: add %eax,%edx # adding 'a' + 'b'

0x08048542 <+80>: mov 0x10(%ebp),%eax

0x08048545 <+83>: mov %eax,0x4(%esp)

0x08048549 <+87>: mov %edx,(%esp)

0x0804854c <+90>: call 0x80484d2 <sophomore>

=> 0x08048551 <+95>: mov -0xc(%ebp),%ecx

0x08048554 <+98>: xor %gs:0x14,%ecx

0x0804855b <+105>: je 0x8048562 <freshman+112>

0x0804855d <+107>: call 0x8048330 <\_\_stack\_chk\_fail@plt>

0x08048562 <+112>: leave

0x08048563 <+113>: ret

And stepping one last time:

(gdb) s

main () at examine\_stack.c:43

43 exit(0);

(gdb) x/32w $esp

0xbffff0c0: 0x0000000c 0x00000019 0x000007dc 0xb7e43a55

0xbffff0d0: 0x000007dc 0x0000000c 0x00000019 0x0000000c

0xbffff0e0: 0x080485c0 0x00000000 0x00000000 0xb7e29905

0xbffff0f0: 0x00000001 0xbffff184 0xbffff18c 0xb7fff000

0xbffff100: 0x00000078 0x00000000 0xb7fdc858 0x00000003

0xbffff110: 0xbffff180 0xb7fc0000 0x00000000 0x00000000

0xbffff120: 0x00000000 0xe6f565fa 0xdc2501ea 0x00000000

0xbffff130: 0x00000000 0x00000000 0x00000001 0x08048370

(gdb) disas main

Dump of assembler code for function main:

0x08048564 <+0>: push %ebp

0x08048565 <+1>: mov %esp,%ebp

0x08048567 <+3>: and $0xfffffff0,%esp

0x0804856a <+6>: sub $0x20,%esp # making room for local variables

0x0804856d <+9>: movl $0x7dc,0x10(%esp) # year

0x08048575 <+17>: movl $0xc,0x14(%esp) # month

0x0804857d <+25>: movl $0x19,0x18(%esp) # day

0x08048585 <+33>: mov 0x10(%esp),%eax

0x08048589 <+37>: mov %eax,0x8(%esp)

0x0804858d <+41>: mov 0x18(%esp),%eax

0x08048591 <+45>: mov %eax,0x4(%esp)

0x08048595 <+49>: mov 0x14(%esp),%eax

0x08048599 <+53>: mov %eax,(%esp)

0x0804859c <+56>: call 0x80484f2 <freshman>

0x080485a1 <+61>: mov %eax,0x1c(%esp) # put result of freshman into 'result' (from %eax)

=> 0x080485a5 <+65>: movl $0x0,(%esp)

0x080485ac <+72>: call 0x8048350 <exit@plt>