

Security Testing Laboratory

Emanuele Viglianisi

Security&Trust Research Unit
Fondazione Bruno Kessler
Trento, Italy
eviglianisi@fbk.eu



Agenda

- Solve Homework-5
- Call stack example
- Exercises
 - Buffer Overflow



Homework-5

Homework-5:

- 1. Download the code from https://github.com/emavgl/sectestlab_2019/tree/master/lab_5/homework-5
- Exploit Integer Overflow vulnerability in order to buy the iPhone for free.
- 3. Try to fix or mitigate the vulnerability



When you call a function, the system sets aside space in memory for that function to do its necessary work.

We call such chunks of memory stack
 frames



When you call a function, the system sets aside space in memory for that function to do its necessary work.

 We call such chunks of memory stack frames

main()



When you call a function, the system sets aside space in memory for that function to do its necessary work.

 We call such chunks of memory stack frames

| 0 | |
|--------|--|
| move() | |
| main() | |



When you call a function, the system sets aside space in memory for that function to do its necessary work.

We call such chunks of memory stack
 frames

| direction() |
|-------------|
| move() |
| main() |



```
#include <stdio.h>
int fool(int a, int b) {
   int c = a + b;
   return c;
}
int main()
{
   fool(1, 2);
   return 0;
}
```

Registries:

- **ESP**: points to the last thing pushed on the stack
- **EIP**: points to the next instruction to execute
- **EBP**: address of the frame's base



```
#include <stdio.h>
int fool(int a, int b) {
   int c = a + b;
   return c;
}
int main()
{
   fool(1, 2);
   return 0;
}
```

Registries:

- **ESP**: points to the last thing pushed on the stack
- EIP: points to the next instruction to execute
- **EBP**: address of the frame's base

CALL <addr>

pushes the current value of **EIP** and changes EIP to <addr>

Arguments are pushed onto the stack before a function call

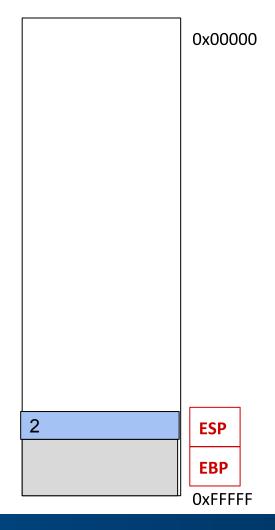


```
// Main's code
// ...
0x00000531 <+13>:
                      push
                              $0x2
0 \times 000000533 < +15 > :
                      push
                              $0x1
0 \times 000000535 < +17>:
                     call 0x4ed <foo1>
0x0000053a <+22>:
                      add
                              $0x8,%esp
   int main()
       foo1(1, 2);
       return 0;
```

0x00000 **ESP EBP**

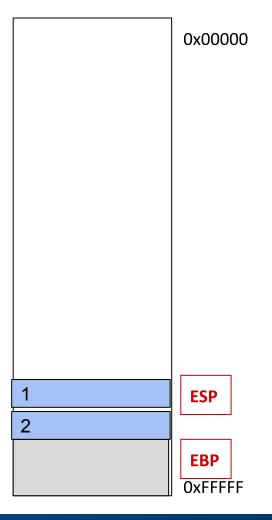


```
// Main's code
// ...
0x00000531 <+13>:
                        push
                                 $0x2
0 \times 000000533 < +15 > :
                        push
                                $0x1EIP
0 \times 000000535 < +17>:
                        call 0x4ed < foo1>
0 \times 00000053a < +22>:
                        add
                                $0x8, %esp
   int main()
       foo1(1, 2);
       return 0;
```





```
// Main's code
// ...
0 \times 000000531 < +13>:
                               $0x2
                       push
0x00000533 <+15>:
                       push
                               $0x1
0 \times 000000535 < +17>:
                      call 0x4ed < foo1 > EIP
0x0000053a <+22>:
                       add
                               $0x8, %esp
   int main()
       foo1(1, 2);
       return 0;
```

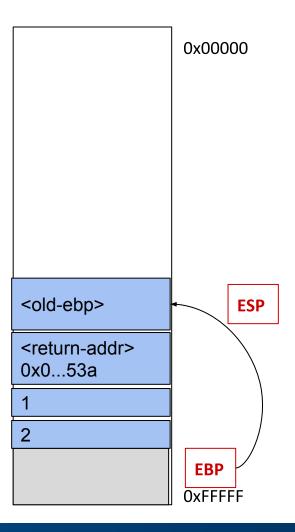




```
0x00000
// Main's code
// ...
0 \times 000000531 < +13>:
                    push
                                $0x2
0 \times 000000533 < +15 > :
                        push
                                $0x1
0x00000535 <+17>:
                     call 0x4ed <foo1>
0 \times 00000053a <+22>:
                                            EIP
   int main()
                                                       <return-addr>
                                                                            ESP
                                                       0x0...53a
       foo1(1, 2);
       return 0;
                                                                            EBP
```



```
// Fool's code
// ...
0 \times 000000504 <+0>:
                           push
                                    %ebp
                                    %esp,%ebp FIP
0 \times 000000505 <+1>:
                           mov
0 \times 000000514 < +16 > :
                                    0x8(%ebp), %edx
                           mov
0 \times 000000517 < +19 > :
                                    0xc(%ebp), %eax
                           mov
0 \times 00000051a < +22>:
                                  %edx, %eax
                           add
                                   %eax,-0x4(%ebp)
0 \times 00000051c < +24>:
                           mov
0 \times 00000051 f <+27>:
                                    -0x4 (%ebp), %eax
                           mov
0 \times 000000522 < +30>:
                           leave
0 \times 000000523 < +31 > :
                           ret
      int fool(int a, int b) {
           int c = a + b;
           return c;
```





```
// Fool's code
// ...
                                                                                  0x00000
0 \times 000000504 <+0>:
                          push
                                   %ebp
0x00000505 <+1>:
                                   %esp,%ebp
                          mov
                                   0x8(%ebp), %edx EIP
0 \times 000000514 < +16 > :
                          mov
                                   0xc(%ebp), %eax
0 \times 000000517 < +19 > :
                          mov
0 \times 00000051a < +22 > :
                          add
                                 %edx,%eax
0 \times 00000051c < +24>:
                                  ext{-0x4} (epp)
                          mov
0 \times 00000051 f <+27>:
                                   -0x4 (%ebp), %eax
                          mov
0 \times 000000522 < +30 > :
                          leave
0 \times 000000523 < +31 > :
                          ret
                                                             <old-ebp>
                                                                                     ESP
                                                                                              EBP
                                                             <return-addr>
      int fool(int a, int b) {
                                                             0x0...53a
           int c = a + b;
           return c;
                                                                                   OxFFFFF
```



```
// Fool's code
// ...
0 \times 000000504 <+0>:
                          push
                                    %ebp
0 \times 000000505 <+1>:
                                    %esp, %ebp
                           mov
0 \times 000000514 < +16 > :
                                    0x8 (%ebp), %edx
                          mov
0x00000517 <+19>:
                                    0xc(%ebp),%eax
                          mov
0 \times 00000051a < +22 > :
                           add
                                   %edx, %eax EIP
0 \times 00000051c < +24>:
                                  ext{-0x4} (epp)
                          mov
0 \times 00000051 f <+27>:
                                    -0x4 (%ebp), %eax
                          mov
0 \times 000000522 < +30 > :
                           leave
0 \times 000000523 < +31 > :
                           ret
      int fool(int a, int b) {
           int c = a + b;
           return c;
```

0x00000 <old-ebp> **ESP EBP** <return-addr> 0x0...53a $EBP + 8 \rightarrow$ $EBX + 12 \rightarrow EAX$ **OxFFFFF**



```
// Fool's code
// ...
                                                                                       0x00000
0 \times 000000504 <+0>:
                           push
                                     %ebp
0 \times 000000505 <+1>:
                                     %esp, %ebp
                           mov
0 \times 000000514 < +16 > :
                                     0x8(%ebp),%edx
                           mov
0 \times 000000517 < +19 > :
                                     0xc(%ebp),%eax
                           mov
0 \times 00000051a < +22>:
                            add
                                     %edx,%eax
                                    %eax, -0x4(%ebp) EIP
0 \times 00000051c < +24>:
                           mov
                                     -0x4 (%ebp), %eax
0 \times 00000051 f <+27>:
                           mov
0 \times 000000522 < +30 > :
                           leave
0 \times 000000523 < +31 > :
                           ret
                                                                <old-ebp>
                                                                                          ESP
                                                                                                   EBP
      int fool(int a, int b) {
                                                                <return-addr>
            int c = a + b;
                                                                0x0...53a
            return c;
                                                                                                                     EAX
                                                                                        EBP + 8 \rightarrow
                                                                                        EBX + 12 \rightarrow EAX
                                                                                                                     0x3
                                                                                       OxFFFFF
                                                                                                                  17
```



```
// Fool's code
// ...
0 \times 000000504 <+0>:
                          push
                                   %ebp
0 \times 000000505 <+1>:
                                   %esp, %ebp
                          mov
0 \times 000000514 < +16 > :
                                   0x8(%ebp),%edx
                          mov
0 \times 000000517 < +19 > :
                                   0xc(%ebp),%eax
                          mov
0 \times 00000051a < +22 > :
                          add
                                   %edx, %eax
0x0000051c <+24>:
                                   %eax,-0x4(%ebp)
                          mov
                                   -0x4(%ebp), %eax
0 \times 00000051 f <+27>:
                          mov
0 \times 000000522 < +30>:
                          leave
                                                             c = 0x3
0 \times 000000523 < +31 > :
                          ret
                                                             <old-ebp>
      int fool(int a, int b) {
                                                             <return-addr>
           int c = a + b;
                                                             0x0...53a
           return c;
```

0x00000

ESP

EBP

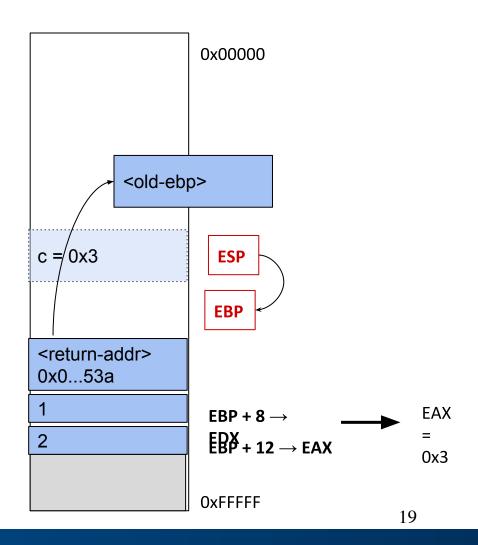
 $EBP + 8 \rightarrow EAX$ = 0x3 0x3

OxFFFFF

18



```
// Fool's code
// ...
0 \times 000000504 <+0>:
                            push
                                     %ebp
0 \times 000000505 <+1>:
                                     %esp, %ebp
                            mov
0 \times 000000514 < +16 > :
                                     0x8(%ebp), %edx
                            mov
0 \times 000000517 < +19 > :
                                     0xc(%ebp),%eax
                            mov
0 \times 00000051a < +22 > :
                            add
                                     %edx, %eax
0 \times 00000051c < +24>:
                                     ext{%eax}, -0x4(ext{%ebp})
                            mov
0 \times 00000051 f <+27>:
                                     -0x4 (%ebp), %eax
                            mov
0 \times 000000522 < +30>:
                            leave
0 \times 000000523 < +31 > :
                            ret
                                        EIP
      int fool(int a, int b) {
            int c = a + b;
            return c;
```





Buffer Overflow

Buffer Overflow (BOF) consists on reading/writing more than the allocated buffer amount



Buffer Overflow

Buffer Overflow (BOF) consists on reading/writing more than the allocated buffer amount

```
int foo() {
    char a;
    char buf[3];
    char password[] = "ciao";
    strcpy(buf, password);
    return 0;
}

int main() {
    foo();
    return 0;
}
OxFFFFF
```

