CSE 410/510 Special Topics: Software Security

Instructor: Dr. Ziming Zhao

Location: Obrian 109

Time: Monday, Wednesday 5:00PM-6:20PM

This Class

- 1. Stack-based buffer overflow
 - a. Place the shellcode at other locations.

This Class

- 1. Stack-based buffer overflow
 - a. Overwrite Saved EBP.
 - b. Defense.

Frame Pointer Attack (Saved EBP/RBP)

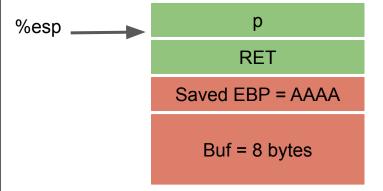
Change the upper level func's return address

```
int vulfoo(char *p)
     char buf[4];
     memcpy(buf, p, 12);
     return 0;
int main(int argc, char *argv[])
     if (argc != 2)
           return 0;
     vulfoo(argv[1]);
```

```
000011cd <vulfoo>:
                                endbr32
  11cd:
            f3 0f 1e fb
  11d1:
            55
                         push %ebp
  11d2:
            89 e5
                                      %esp,%ebp
                                mov
  11d4:
            53
                         push %ebx
  11d5:
            83 ec 04
                                sub $0x4,%esp
  11d8:
            e8 58 00 00 00
                                call 1235 <__x86.get_pc_thunk.ax>
  11dd:
            05 fb 2d 00 00
                                add $0x2dfb,%eax
  11e2:
                         push $0xc
            6a 0c
  11e4:
            ff 75 08
                                pushl 0x8(%ebp)
  11e7:
            8d 55 f8
                                lea -0x8(%ebp),%edx
                         push %edx
  11ea:
            52
  11eb:
            89 c3
                               %eax,%ebx
                         mov
  11ed:
            e8 7e fe ff ff
                                call 1070 <memcpy@plt>
  11f2:
            83 c4 0c
                                add $0xc,%esp
  11f5:
            b8 00 00 00 00
                                mov $0x0,%eax
  11fa:8b 5d fc
                               -0x4(%ebp),%ebx
                         mov
  11fd:
                         leave
  11fe:c3
                   ret
```

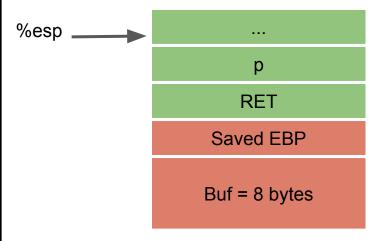
p
RET
Saved EBP
Buf = 8 bytes

```
000011cd <vulfoo>:
  11cd:
            f3 0f 1e fb
                                endbr32
  11d1:
            55
                         push %ebp
  11d2:
            89 e5
                                      %esp,%ebp
                                mov
  11d4:
            53
                          push %ebx
  11d5:
            83 ec 04
                                sub
                                     $0x4,%esp
  11d8:
            e8 58 00 00 00
                                call
                                    1235 <__x86.get_pc_thunk.ax>
  11dd:
            05 fb 2d 00 00
                                add
                                     $0x2dfb,%eax
  11e2:
            6a 0c
                          push $0xc
  11e4:
            ff 75 08
                                pushl 0x8(%ebp)
  11e7:
            8d 55 f8
                                lea -0x8(%ebp),%edx
                         push %edx
  11ea:
            52
  11eb:
            89 c3
                               %eax,%ebx
                          mov
                                call 1070 <memcpy@plt>
  11ed:
            e8 7e fe ff ff
  11f2:
            83 c4 0c
                                add
                                     $0xc,%esp
  11f5:
            b8 00 00 00 00
                                mov $0x0,%eax
  11fa:8b 5d fc
                               -0x4(%ebp),%ebx
                         mov
  11fd:
            c9
                         leave
  11fe:c3
                   ret
```

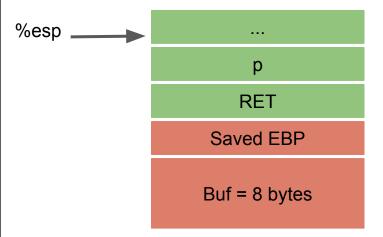


$$\%$$
ebp = AAAA

000011ff <r< th=""><th></th><th></th><th></th></r<>			
11ff: f3 0f	f 1e fb	endbr	
1203:	55	push	%ebp
1204:	89 e5		mov %esp,%ebp
1206:	e8 2a 00 00	00 (call 1235 <x86.get_pc_thunk.ax></x86.get_pc_thunk.ax>
120b:	05 cd 2d 00	00	add \$0x2dcd,%eax
1210:	83 7d 08 02	<u>)</u>	cmpl \$0x2,0x8(%ebp)
1214:	74 07		je 121d <main+0x1e></main+0x1e>
1216:	b8 00 00 00	00	mov \$0x0,%eax
121b:	eb 16		jmp 1233 <main+0x34></main+0x34>
121d:	8b 45 0c		mov 0xc(%ebp),%eax
1220:	83 c0 04		add \$0x4,%eax
1223:	8b 00		mov (%eax),%eax
1225:	50	push	%eax
1226:	e8 a2 ff ff f	f	call 11cd <vulfoo></vulfoo>
122b:	83 c4 04		add \$0x4,%esp
122e:	00 00 8d	00	mov \$0x0,%eax
1233:	c9	leave	
1234:	c3	ret	



000011ff <m< th=""><th>ain>:</th><th></th><th></th></m<>	ain>:		
11ff: f3 0f 1	1e fb	endbr	r32
1203:	55	push	%ebp
1204:	89 e5		mov %esp,%ebp
1206:	e8 2a 00 00 00	0	call 1235 <x86.get_pc_thunk.ax></x86.get_pc_thunk.ax>
120b:	05 cd 2d 00 0	0	add \$0x2dcd,%eax
1210:	83 7d 08 02		cmpl
1214:	74 07		je 121d <main+0x1e></main+0x1e>
1216:	b8 00 00 00 0	0	mov \$0x0,%eax
121b:	eb 16		jmp 1233 <main+0x34></main+0x34>
121d:	8b 45 0c		mov 0xc(%ebp),%eax
1220:	83 c0 04		add \$0x4,%eax
1223:	8b 00		mov (%eax),%eax
1225:	50	push	%eax
1226:	e8 a2 ff ff ff		call 11cd <vulfoo></vulfoo>
122b:	83 c4 04		add \$0x4,%esp
122e:	b8 00 00 00 0	0	mov \$0x0,%eax
1233:	c9	leave	!
1234:	c3	ret	



```
000011ff <main>:
  11ff: f3 0f 1e fb
                         endbr32
  1203:
                         push %ebp
  1204:
            89 e5
                                      %esp,%ebp
                                mov
  1206:
            e8 2a 00 00 00
                                call 1235 <__x86.get_pc_thunk.ax>
  120b:
            05 cd 2d 00 00
                                add $0x2dcd,%eax
  1210:
            83 7d 08 02
                                cmpl $0x2,0x8(%ebp)
  1214:
            74 07
                                   121d <main+0x1e>
  1216:
            b8 00 00 00 00
                                mov $0x0,%eax
                                jmp 1233 <main+0x34>
  121b:
            eb 16
  121d:
            8b 45 0c
                                mov 0xc(%ebp),%eax
  1220:
            83 c0 04
                                add $0x4.%eax
  1223:
            8b 00
                                mov (%eax),%eax
                         push %eax
  1225:
             50
                                call 11cd <vulfoo>
  1226:
            e8 a2 ff ff ff
  122b:
            83 c4 04
                                add $0x4,%esp
  122e:
            b8 00 00 00 00
                                mov $0x0.%eax
  1233:
            c9
                         leave
            с3
  1234:
                         ret
```

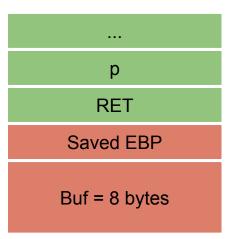
mov %ebp, %esp

pop %ebp

metric properties and the second seco

- 1. %esp = AAAA
- 2. %ebp = *(AAAA); %esp += 4, AAAE

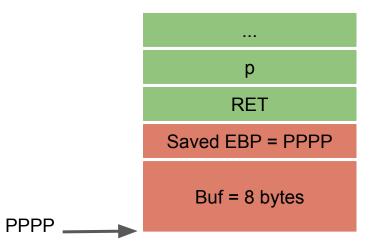
```
000011ff <main>:
  11ff: f3 0f 1e fb
                         endbr32
  1203:
             55
                         push %ebp
  1204:
            89 e5
                                      %esp,%ebp
                                mov
  1206:
            e8 2a 00 00 00
                                call 1235 <__x86.get_pc_thunk.ax>
                                add $0x2dcd,%eax
  120b:
            05 cd 2d 00 00
  1210:
            83 7d 08 02
                                cmpl $0x2,0x8(%ebp)
  1214:
            74 07
                                    121d <main+0x1e>
            b8 00 00 00 00
                                mov $0x0,%eax
  1216:
                                jmp 1233 <main+0x34>
  121b:
            eb 16
  121d:
            8b 45 0c
                                mov 0xc(%ebp),%eax
  1220:
            83 c0 04
                                add $0x4.%eax
  1223:
            8b 00
                                      (%eax),%eax
                                mov
                         push %eax
  1225:
             50
                                call 11cd <vulfoo>
  1226:
            e8 a2 ff ff ff
  122b:
            83 c4 04
                                add
                                     $0x4,%esp
  122e:
            b8 00 00 00 00
                                      $0x0,%eax
                                mov
  1233:
            c9
                         leave
            с3
  1234:
                         ret
```



1. %eip = *(AAAE)

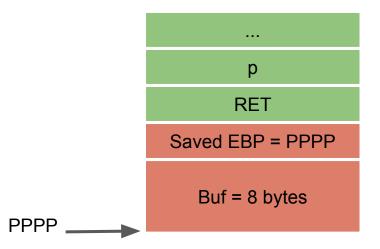
Overflow6 32bit: If we know where is buf is

000011ff <main>:</main>					
11ff: f3 0f 1e fb		endbr	32		
1203:	55	push	%ebp		
1204:	89 e5		mov %esp,%ebp		
1206:	e8 2a 00 00 00)	call 1235 <x86.get_pc_thunk.ax></x86.get_pc_thunk.ax>		
120b:	05 cd 2d 00 00)	add \$0x2dcd,%eax		
1210:	83 7d 08 02		cmpl		
1214:	74 07		je 121d <main+0x1e></main+0x1e>		
1216:	b8 00 00 00 00	0	mov \$0x0,%eax		
121b:	eb 16		jmp 1233 <main+0x34></main+0x34>		
121d:	8b 45 0c		mov 0xc(%ebp),%eax		
1220:	83 c0 04		add \$0x4,%eax		
1223:	8b 00		mov (%eax),%eax		
1225:	50	push	%eax		
1226:	e8 a2 ff ff ff		call 11cd <vulfoo></vulfoo>		
122b:	83 c4 04		add \$0x4,%esp		
122e:	b8 00 00 00 00	0	mov \$0x0,%eax		
1233:	c9	leave			
1234:	c3	ret			



Overflow6 32bit: If we don't know where is buf is

000011ff <main>:</main>					
11ff: f3 0f	1e fb	endbr	·32		
1203:	55	push	%ebp		
1204:	89 e5		mov %esp,%ebp		
1206:	e8 2a 00 00 00	0	call 1235 <x86.get_pc_thunk.ax></x86.get_pc_thunk.ax>		
120b:	05 cd 2d 00 0	0	add \$0x2dcd,%eax		
1210:	83 7d 08 02		cmpl \$0x2,0x8(%ebp)		
1214:	74 07		je 121d <main+0x1e></main+0x1e>		
1216:	b8 00 00 00 0	0	mov \$0x0,%eax		
121b:	eb 16		jmp 1233 <main+0x34></main+0x34>		
121d:	8b 45 0c		mov 0xc(%ebp),%eax		
1220:	83 c0 04		add \$0x4,%eax		
1223:	8b 00		mov (%eax),%eax		
1225:	50	push	%eax		
1226:	e8 a2 ff ff ff		call 11cd <vulfoo></vulfoo>		
122b:	83 c4 04		add \$0x4,%esp		
122e:	b8 00 00 00 0	0	mov \$0x0,%eax		
1233:	c9	leave			
1234:	c3	ret			



Conditions we depend on to pull off the attack of returning to shellcode on stack

- 1. The ability to put the shellcode onto stack (env, command line)
- 2. The stack is executable
- 3. The ability to overwrite RET addr on stack before instruction **ret** is executed or to overwrite Saved EBP
- 4. Know the address of the destination function