Pointers and I/O

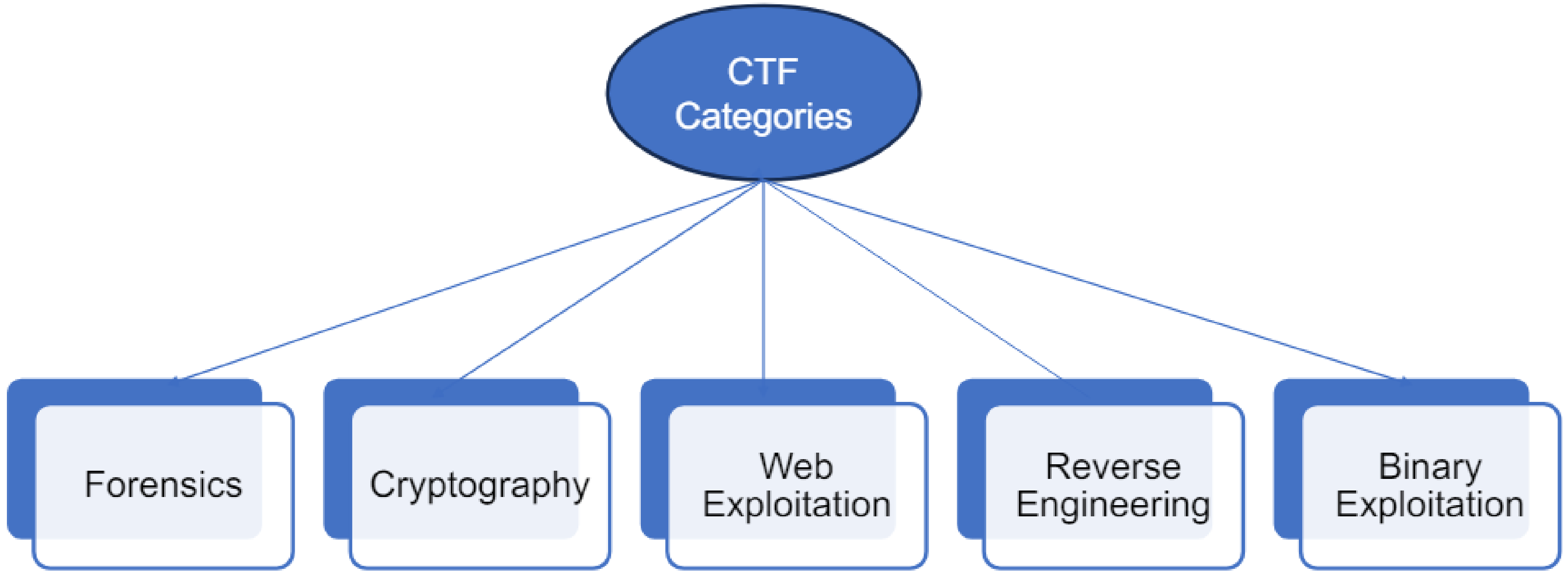
Ali Ghaffarian

Table of Contents

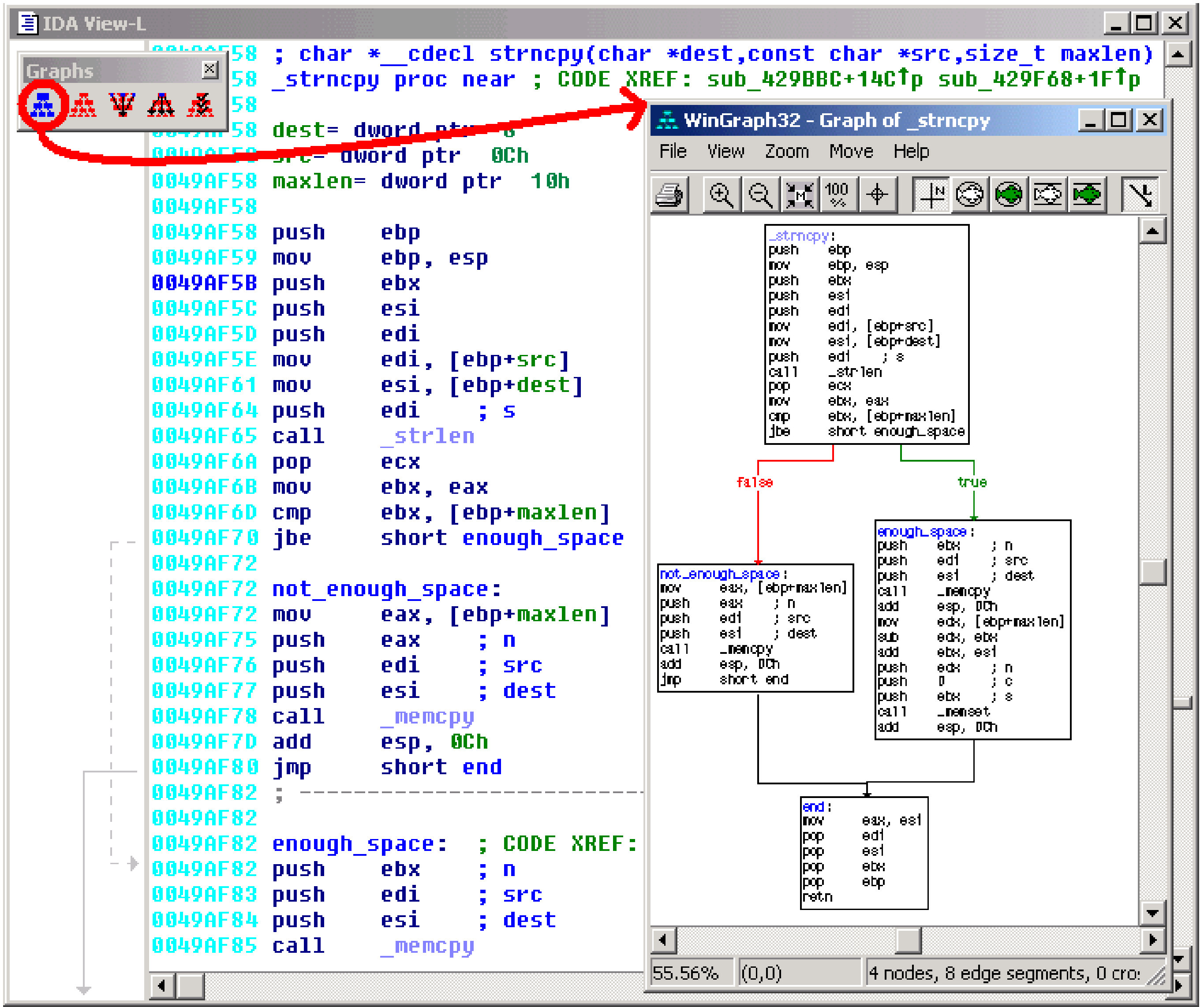
Summary of Last Talk

ctf_
CAPTURE THE FLAG

Categories


CTF Categories
Forensics
Cryptography
Web Exploitation
Reverse Engineering
Binary Exploitation

Reverse Engineering


IDA View-L
Graphs
WinGraph32 - Graph of _strncpy
File View Zoom Move Help
55.56%
(0,0)
4 nodes, 8 edge segments, 0 cro

```
        58 ; char *__cdecl strncpy(char *dest,const char *src,size_t maxlen)
        58 _strncpy proc near ; CODE XREF: sub_429BBC+14C↑p sub_429F68+1F↑p
        58
        58 dest= dword ptr  8
0049AF58 src= dword ptr  0Ch
0049AF58 maxlen= dword ptr  10h
0049AF58
0049AF58 push    ebp
0049AF59 mov     ebp, esp
0049AF5B push    ebx
0049AF5C push    esi
0049AF5D push    edi
0049AF5E mov     edi, [ebp+src]
0049AF61 mov     esi, [ebp+dest]
0049AF64 push    edi     ; s
0049AF65 call    _strlen
0049AF6A pop     ecx
0049AF6B mov     ebx, eax
0049AF6D cmp     ebx, [ebp+maxlen]
0049AF70 jbe     short enough_space
0049AF72
0049AF72 not_enough_space:
0049AF72 mov     eax, [ebp+maxlen]
0049AF75 push    eax     ; n
0049AF76 push    edi     ; src
0049AF77 push    esi     ; dest
0049AF78 call    _memcpy
0049AF7D add     esp, 0Ch
0049AF80 jmp     short end
0049AF82 ; ---------------------------------
0049AF82
0049AF82 enough_space:  ; CODE XREF:
0049AF82 push    ebx     ; n
0049AF83 push    edi     ; src
0049AF84 push    esi     ; dest
0049AF85 call    _memcpy
```

```
_strncpy:
push    ebp
mov     ebp, esp
push    ebx
push    esi
push    edi
mov     edi, [ebp+src]
mov     esi, [ebp+dest]
push    edi     ; s
call    _strlen
pop     ecx
mov     ebx, eax
cmp     ebx, [ebp+maxlen]
jbe     short enough_space
```

false

```
not_enough_space:
mov     eax, [ebp+maxlen]
push    eax     ; n
push    edi     ; src
push    esi     ; dest
call    _memcpy
add     esp, 0Ch
jmp     short end
```

true

```
enough_space:
push    ebx     ; n
push    edi     ; src
push    esi     ; dest
call    _memcpy
add     esp, 0Ch
mov     edx, [ebp+maxlen]
sub     edx, ebx
add     ebx, esi
push    edx     ; n
push    0       ; c
push    ebx     ; s
call    _memset
add     esp, 0Ch
```

```
end:
mov     eax, esi
pop     edi
pop     esi
pop     ebx
pop     ebp
retn
```

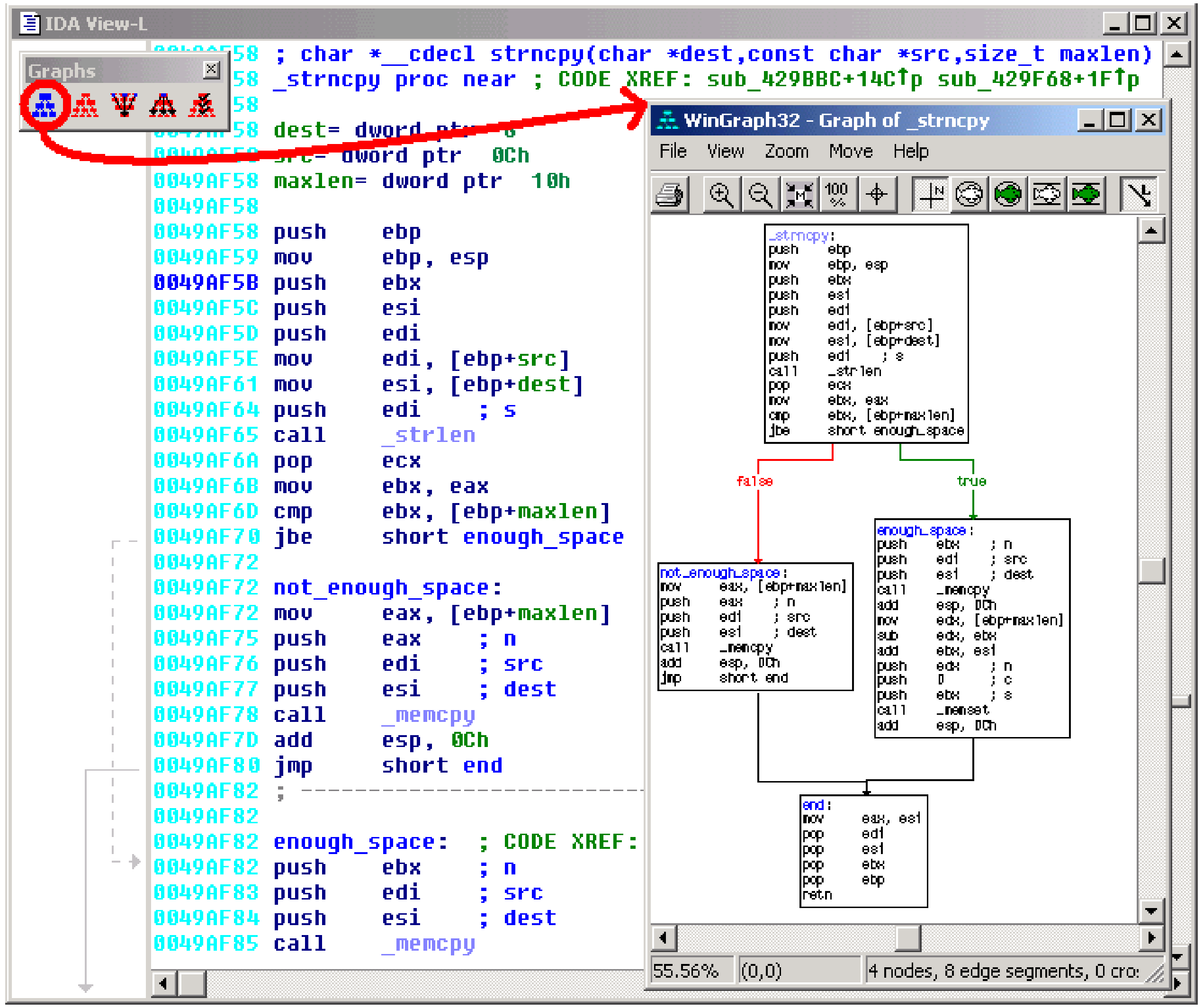
Instances

- Game Hacking
- Malware Analysis
- Cracking

Types of Analysis

Static vs Dynamic

Static Analysis


IDA View-L
Graphs
WinGraph32 - Graph of _strncpy
File View Zoom Move Help
false
true
55.56%
(0,0)
4 nodes, 8 edge segments, 0 cro

```
0049AF58 ; char *__cdecl strncpy(char *dest,const char *src,size_t maxlen)
0049AF58 _strncpy proc near ; CODE XREF: sub_429BBC+14C↑p sub_429F68+1F↑p
0049AF58
0049AF58 dest= dword ptr  8
0049AF58 src= dword ptr  0Ch
0049AF58 maxlen= dword ptr  10h
0049AF58
0049AF58 push    ebp
0049AF59 mov     ebp, esp
0049AF5B push    ebx
0049AF5C push    esi
0049AF5D push    edi
0049AF5E mov     edi, [ebp+src]
0049AF61 mov     esi, [ebp+dest]
0049AF64 push    edi     ; s
0049AF65 call    _strlen
0049AF6A pop     ecx
0049AF6B mov     ebx, eax
0049AF6D cmp     ebx, [ebp+maxlen]
0049AF70 jbe     short enough_space
0049AF72
0049AF72 not_enough_space:
0049AF72 mov     eax, [ebp+maxlen]
0049AF75 push    eax     ; n
0049AF76 push    edi     ; src
0049AF77 push    esi     ; dest
0049AF78 call    _memcpy
0049AF7D add     esp, 0Ch
0049AF80 jmp     short end
0049AF82 ; ---------------------------------------
0049AF82
0049AF82 enough_space:   ; CODE XREF:
0049AF82 push    ebx     ; n
0049AF83 push    edi     ; src
0049AF84 push    esi     ; dest
0049AF85 call    _memcpy
```

```
_strncpy:
push    ebp
mov     ebp, esp
push    ebx
push    esi
push    edi
mov     edi, [ebp+src]
mov     esi, [ebp+dest]
push    edi     ; s
call    _strlen
pop     ecx
mov     ebx, eax
cmp     ebx, [ebp+maxlen]
jbe     short enough_space
```

```
not_enough_space:
mov     eax, [ebp+maxlen]
push    eax     ; n
push    edi     ; src
push    esi     ; dest
call    _memcpy
add     esp, 0Ch
jmp     short end
```

```
enough_space:
push    ebx     ; n
push    edi     ; src
push    esi     ; dest
call    _memcpy
add     esp, 0Ch
mov     edx, [ebp+maxlen]
sub     edx, ebx
add     ebx, esi
push    edx     ; n
push    0       ; c
push    ebx     ; s
call    _memset
add     esp, 0Ch
```

```
end:
mov     eax, esi
pop     edi
pop     esi
pop     ebx
pop     ebp
retn
```

Tools

- IDA
- Ghidra
- Dogbolt.org

Pointers

- Address of Memory

```
int number = 0x55223344;
int *ptr_number = &number;
void *v_ptr_number = &number;
char *c_ptr_number = &number;
```

Pointer Types

		??	0x0000009CF0FFF86C
	(ptr_number)	0x0000009CF0FFF860 (8B)	0x0000009CF0FFF874
	(v_ptr_number)	0x0000009CF0FFF860 (8B)	0x0000009CF0FFF86C
	(c_ptr_number)	0x0000009CF0FFF860 (8B)	0x0000009CF0FFF864
		0x55	0x0000009CF0FFF863
	ptr_number	0x22	0x0000009CF0FFF862
		0x33	0x0000009CF0FFF861
c_ptr_number		0x44	0x0000009CF0FFF860

Demo

ptr.c, memory_view.c

Input/Output

File Descriptors

- A handle for a I/O resource
- Local to your program

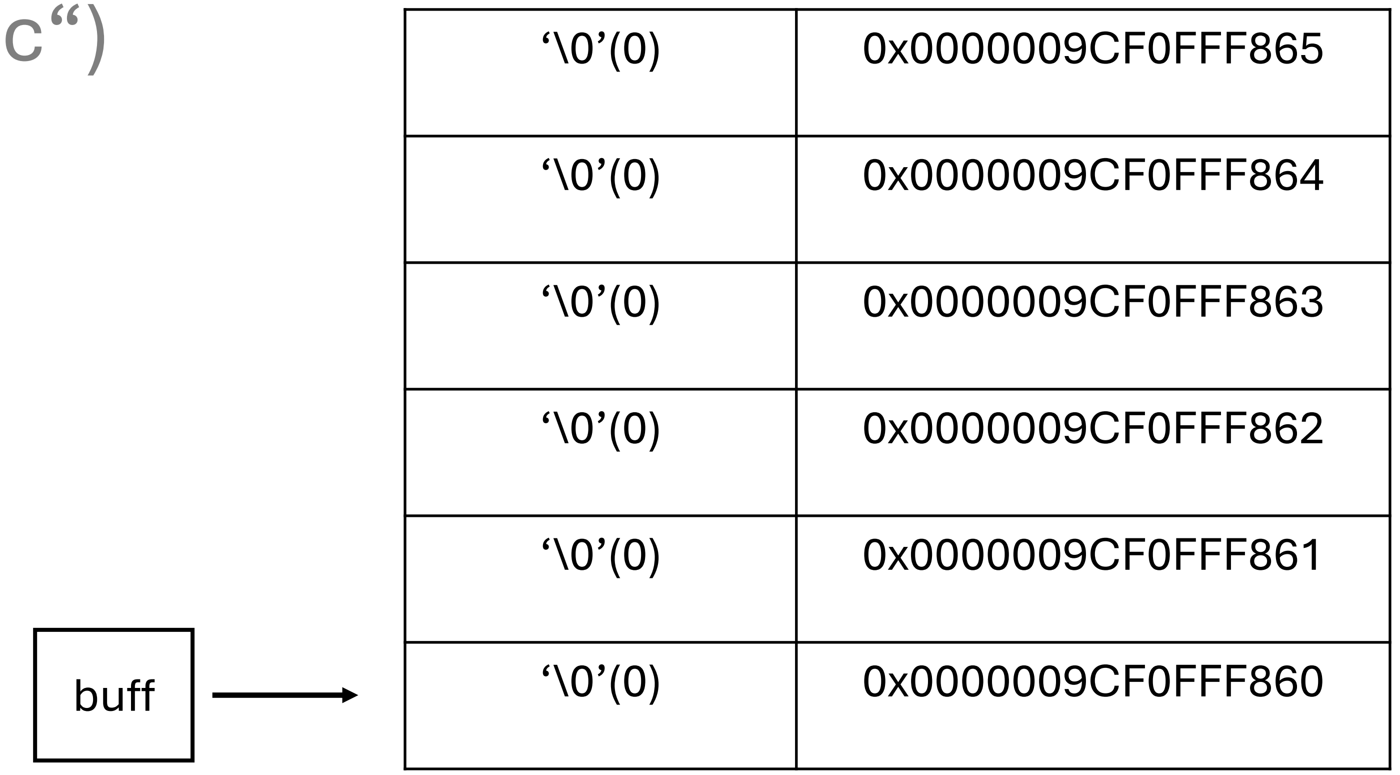
FD’s

```
#include <fcntl.h>

int main(){
	char buff[128] = {0};
	int fd = open("./hello_world.txt", O_RDONLY, 0);
	int read_bytes = read(fd, buff, sizeof(buff));
	write(1 , buff, read_bytes);
}
```

File descriptor	offset	path	permissions
0	0	[STDIN]	Read only
1	0	[STDOUT]	Write only
2	0	[STDERR]	Write only

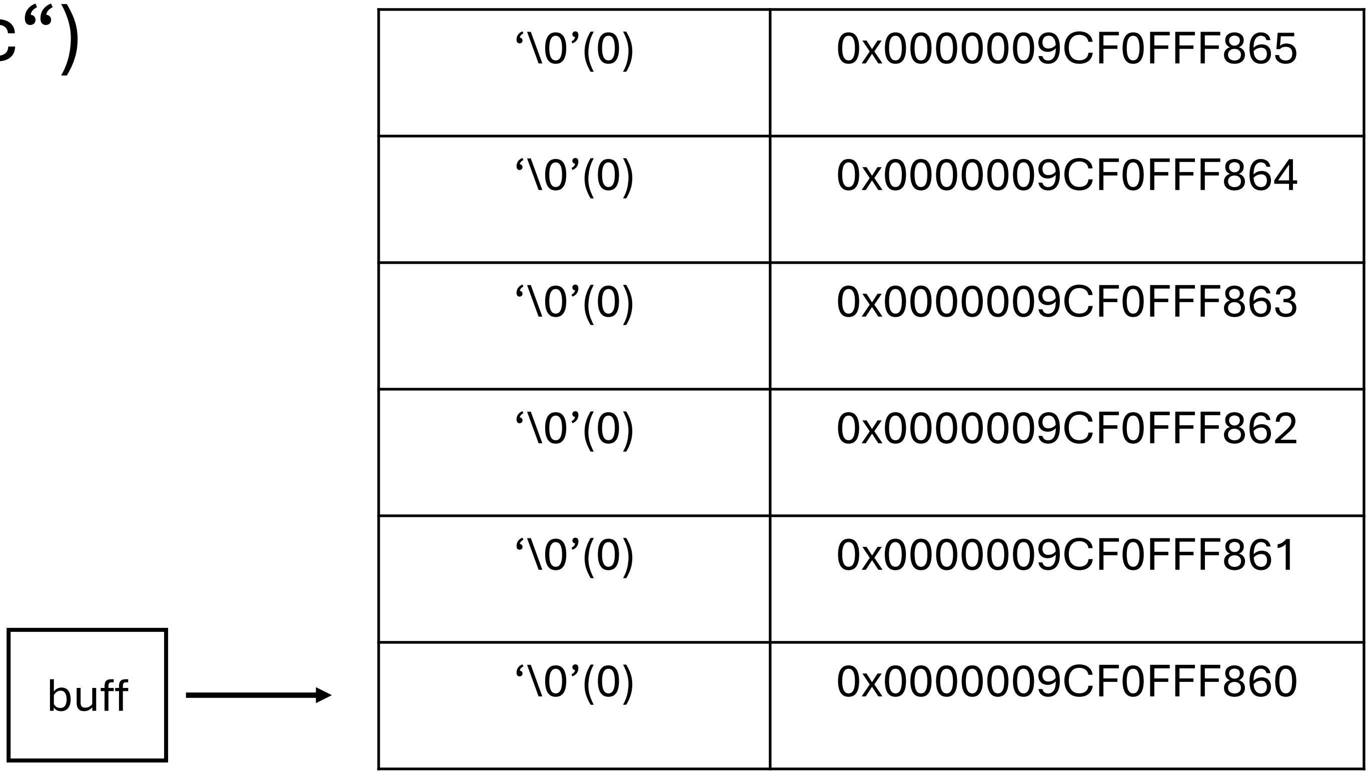
1. open(“./hello_world.c“)
2. read(3)
3. write(1)
4. close(3)



‘\0’(0)	0x0000009CF0FFF865
‘\0’(0)	0x0000009CF0FFF864
‘\0’(0)	0x0000009CF0FFF863
‘\0’(0)	0x0000009CF0FFF862
‘\0’(0)	0x0000009CF0FFF861
‘\0’(0)	0x0000009CF0FFF860

File descriptor	offset	path	permissions
0	0	[STDIN]	Read only
1	0	[STDOUT]	Write only
2	0	[STDERR]	Write only
3	0	./hello_world.c	Read only

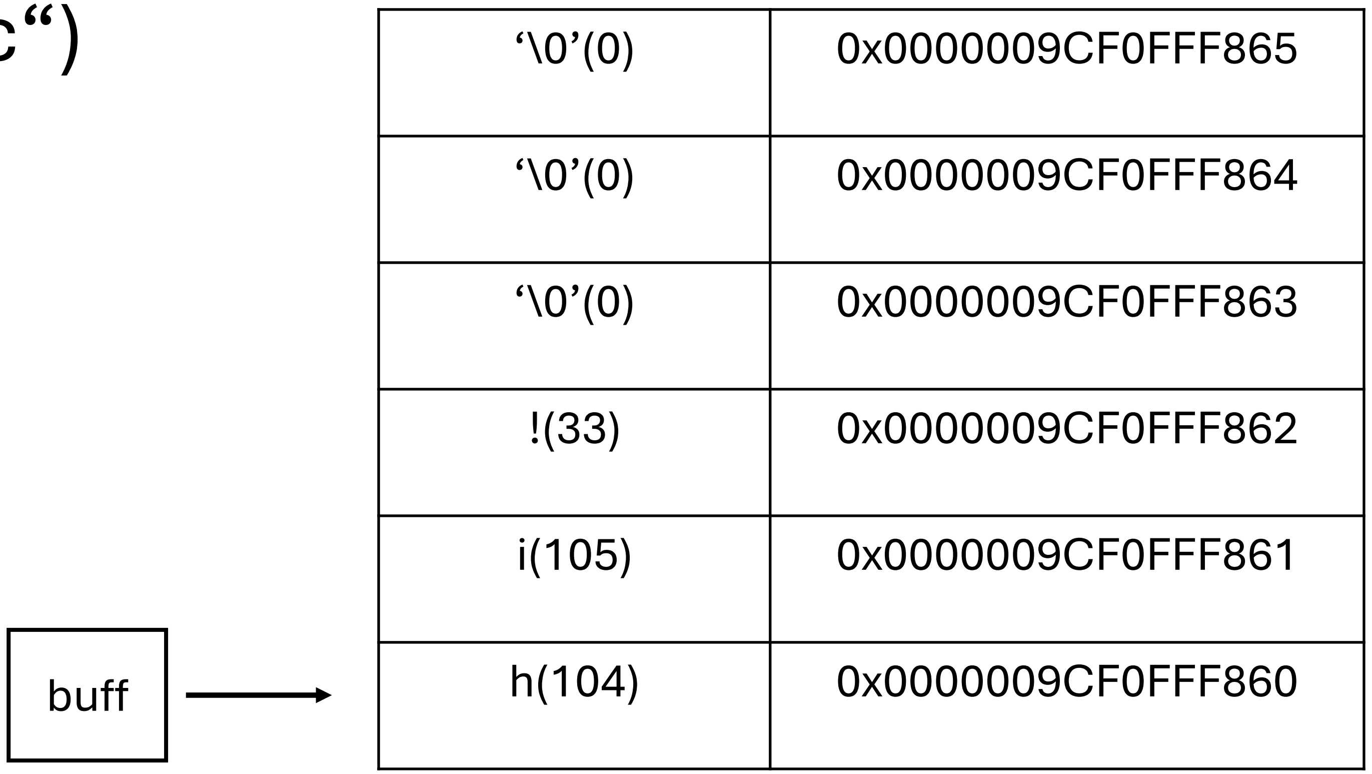
1. open(“./hello_world.c“)
2. read(3)
3. write(1)
4. close(3)



‘\0’(0)	0x0000009CF0FFF865
‘\0’(0)	0x0000009CF0FFF864
‘\0’(0)	0x0000009CF0FFF863
‘\0’(0)	0x0000009CF0FFF862
‘\0’(0)	0x0000009CF0FFF861
‘\0’(0)	0x0000009CF0FFF860

File descriptor	offset	path	permissions
0	0	[STDIN]	Read only
1	0	[STDOUT]	Write only
2	0	[STDERR]	Write only
3	3	./hello_world.c	Read only

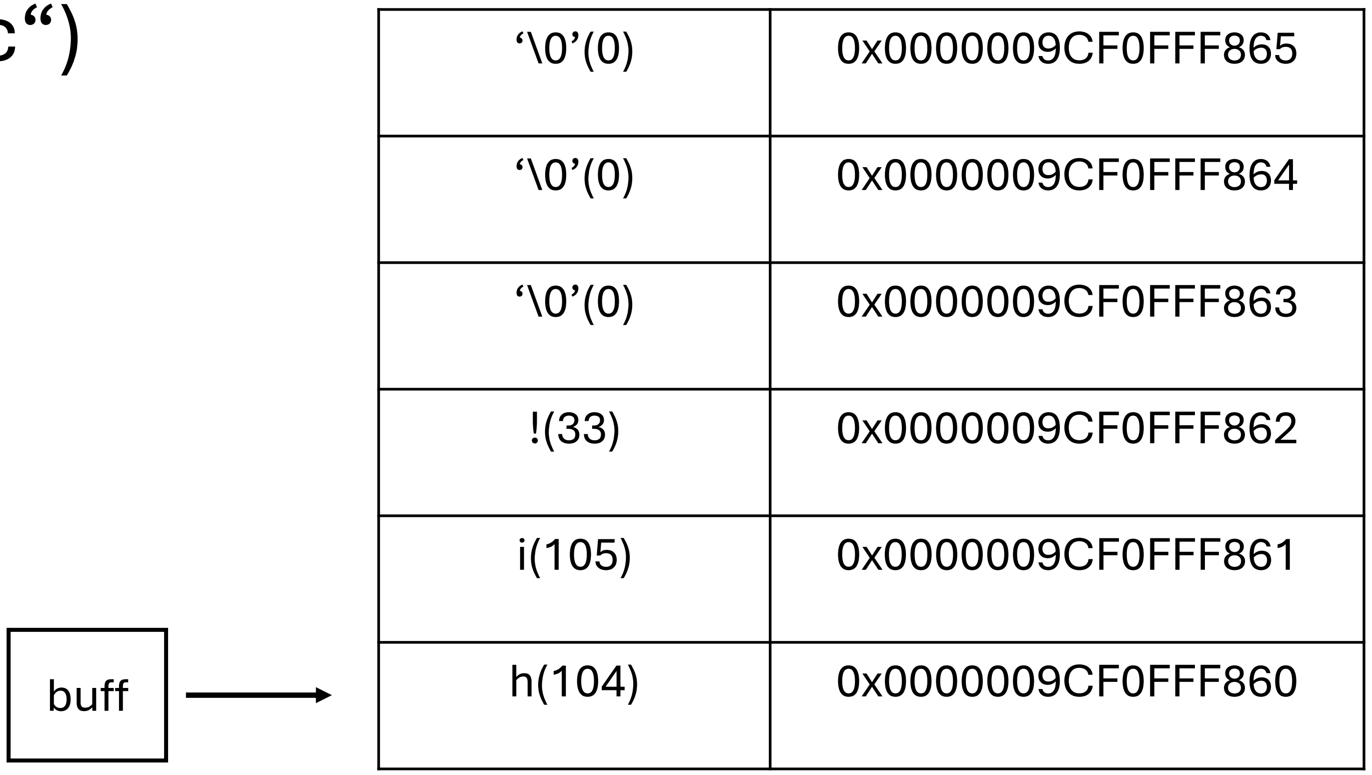
1. open(“./hello_world.c“)
2. read(3)
3. write(1)
4. close(3)



‘\0’(0)	0x0000009CF0FFF865
‘\0’(0)	0x0000009CF0FFF864
‘\0’(0)	0x0000009CF0FFF863
!(33)	0x0000009CF0FFF862
i(105)	0x0000009CF0FFF861
h(104)	0x0000009CF0FFF860

File descriptor	offset	path	permissions
0	0	[STDIN]	Read only
1	0	[STDOUT]	Write only
2	0	[STDERR]	Write only
3	3	./hello_world.c	Read only

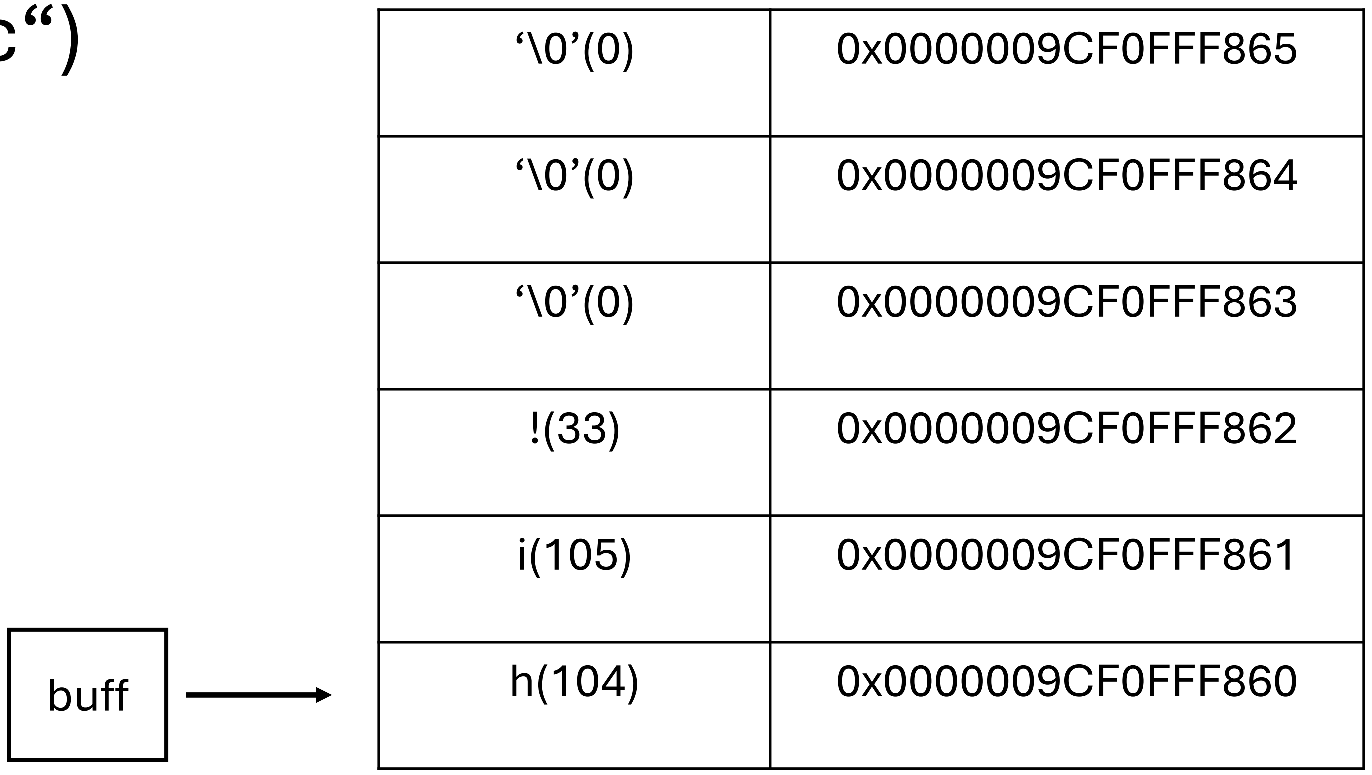
1. open(“./hello_world.c“)
2. read(3)
3. write(1)
4. close(3)



‘\0’(0)	0x0000009CF0FFF865
‘\0’(0)	0x0000009CF0FFF864
‘\0’(0)	0x0000009CF0FFF863
!(33)	0x0000009CF0FFF862
i(105)	0x0000009CF0FFF861
h(104)	0x0000009CF0FFF860

File descriptor	offset	path	permissions
0	0	[STDIN]	Read only
1	0	[STDOUT]	Write only
2	0	[STDERR]	Write only

1. open(“./hello_world.c“)
2. read(3)
3. write(1)
4. close(3)



‘\0’(0)	0x0000009CF0FFF865
‘\0’(0)	0x0000009CF0FFF864
‘\0’(0)	0x0000009CF0FFF863
!(33)	0x0000009CF0FFF862
i(105)	0x0000009CF0FFF861
h(104)	0x0000009CF0FFF860

What about stdio?

Demo

More Reversing

The Compilation Process

1. Compile
2. Assemble
3. Link

```
#include <stdio.h>

int main(){
	printf("hello world!\n");
}
```

Compile

hello_world.s

```
lea     rax, .LC0[rip]
mov     rcx, rax
call    puts
mov     eax, 0
add     rsp, 32
pop     rbp
ret
```

Assemble

hello_world.o

```
Contents of section .text:
 0000 554889e5 4883ec20 e8000000 00488d05  UH..H.. .....H..
 0010 00000000 4889c1e8 00000000 b8000000  ....H...........
 0020 004883c4 205dc390 90909090 90909090  .H.. ]..........
```

Link

```
$dir

10/20/2025  05:38 PM          128,829      hello_world.exe
10/17/2025  05:55 PM          66           hello_world.c
10/20/2025  05:36 PM          882          hello_world.o
10/20/2025  05:34 PM          571          hello_world.s
```

Know Your Compiler!

So Much Added Code!

```
int WinMainCRTStartup(void);
int mainCRTStartup(void);
int32_t atexit(void (*func)(void));
int64_t __gcc_register_frame(void);
void __gcc_deregister_frame(void) __pure;
int64_t main(void);
```

Find the Relevant Code

```
int64_t main() {
	__main();
	puts("hello world!");
	return 0;
}
```