

2025.09.17

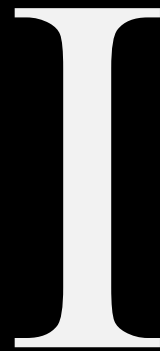
# Buffer OverFlow

가디언 시스템 보안 & 취약점 분석 세미나 2.1

Guardian 2025

# Shell Code

as ASM



# 목표

프로그램의 흐름을 탈취하자

## 원하는 코드를 실행하고 싶다

## 가장 좋은 도구는?

윈도우는 PowerShell, 리눅스는 Shell(/bin/shell)

# Shell

/bin/sh

```
zirajs@ubuntu:~$ sh
$
zirajs@ubuntu:~$ /bin/sh
$ /bin/sh
$
$
```

셸도 프로그램이므로 nano {file} 같은 식으로 실행 가능하다

셸 안에서 셸 실행도 가능

# Syscall

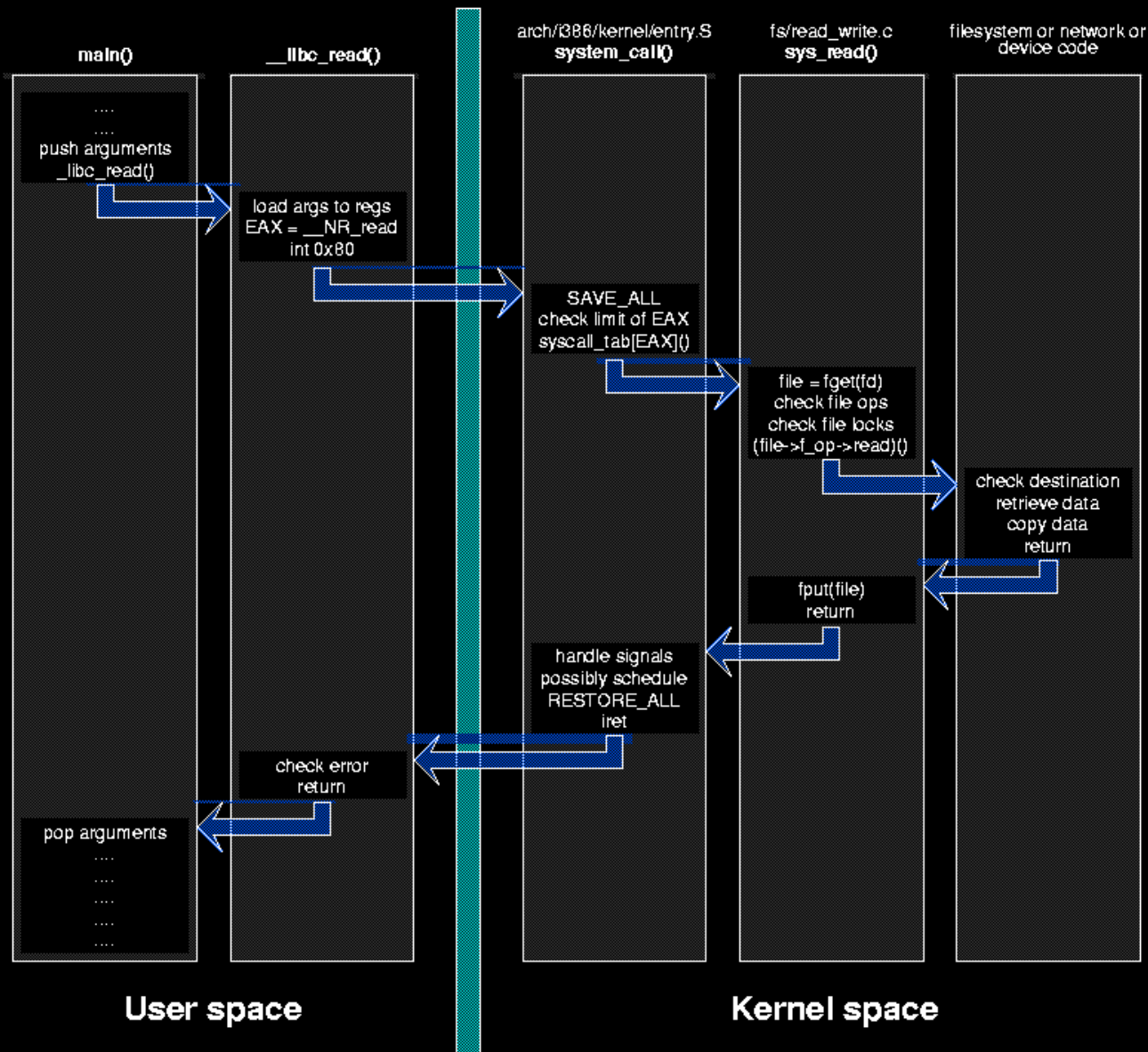
커널에 필요한 기능 요청

프로그램이 read, write 하고 싶음

근데 그건 권한 없는데?

Syscall로 직접 호출하자

# Syscall



시스템한테 “해줘” 하는 거임  
도라에몽 나 이거 출력해줘

# Syscall

커널에 필요한 기능 요청

겁나 많다.

필요할때 찾아보자

reference

[chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md](https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md)

Guardian 2025

%rax	System call	%rdi	%rsi	%rdx	%r10	%r8	%r9
0	sys_read	unsigned int fd	char *buf	size_t count			
1	sys_write	unsigned int fd	const char *buf	size_t count			
2	sys_open	%rax	System call	%rdi	%rsi	%rdx	%r10
3	sys_close	0	sys_read	unsigned int fd	char *buf	size_t count	
4	sys_stat	1	sys_write	unsigned int fd	const char *buf	size_t count	
5	sys_fstat	2	sys_open	const char *filename	int flags	int mode	
6	sys_lstat	3	sys_close	unsigned int fd			
7	sys_poll	4	sys_stat	const char *filename	struct stat *statbuf		
8	sys_lseek	5	sys_fstat	unsigned int fd	struct stat *statbuf		
9	sys_mmap	6	sys_lstat	lconst char *filename	struct stat *statbuf		
10	sys_mprotect	7	sys_poll	struct poll_fd *ufds	unsigned int nfds	long timeout_msecs	
11	sys_munmap	8	sys_lseek	unsigned int fd	off_t offset	unsigned int origin	
12	sys_brk	9	sys_mmap	unsigned long addr	unsigned long len	unsigned long prot	unsigned long flags
13	sys_rt_sigaction	10	sys_mprotect	unsigned long start	size_t len	unsigned long prot	
14	sys_rt_sigprocmask	11	sys_munmap	unsigned long addr	size_t len		
15	sys_rt_sigreturn	12	sys_brk	unsigned long brk			
16	sys_ioctl	13	sys_rt_sigaction	int sig	const struct sigaction *act	struct sigaction *oact	size_t sigsetsize
17	sys_pread64	14	sys_rt_sigprocmask	int how	sigset_t *nset	sigset_t *oset	size_t sigsetsize
18	sys_pwrite64	15	sys_rt_sigreturn	unsigned long _unused			
19	sys_readv	16	sys_ioctl	unsigned int fd	unsigned int cmd	unsigned long arg	
20	sys_writev	17	sys_pread64	unsigned long fd	char *buf	size_t count	loff_t pos
21	sys_access	18	sys_pwrite64	unsigned int fd	const char *buf	size_t count	loff_t pos
22	sys_pipe	19	sys_readv	unsigned long fd	const struct iovec *vec	unsigned long vlen	
23	sys_select	20	sys_writev	unsigned long fd	const struct iovec *vec	unsigned long vlen	
24	sys_sched_yield	21	sys_access	const char *filename	int mode		
25	sys_mremap	22	sys_pipe	int *filedes			
26	sys_msync	23	sys_select	int n	fd_set *inp	fd_set *outp	fd_set *exp
27	sys_mincore	24	sys_sched_yield				
28	sys_madvise	25	sys_mincore	unsigned long addr	unsigned long old_len	unsigned long new_len	unsigned long flags
29	sys_shmget	26	sys_msync	unsigned long start	size_t len	int flags	
		27	sys_mincore	unsigned long start	size_t len	unsigned char *vec	

# 실습 [shell\_basic]

[dreamhack.io/wargame/challenges/410](https://dreamhack.io/wargame/challenges/410)

CTF에서 적어도 한 문제는 나오는 클래식



# System()

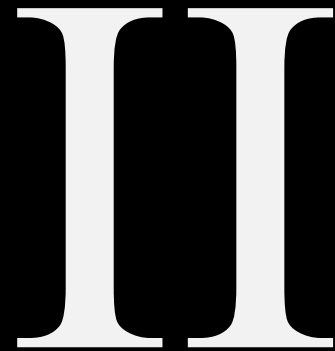
System("/bin/sh")

## Shell로 child process를 생성

execl("/bin/sh", "sh", "-c", command, (char \*) NULL);

# Buffer OverFlow

Hello, world!



# Ret / Leave

function call

```
[ret]  
pop rip;
```

```
[leave]  
mov rsp, rbp;  
pop rbp;
```

# Ret2Func

gets(), strcpy() = Shell

RBP

Prog. Stack

Return Addr

SPO

RBP- 0x20 =

Buffer [0x20]

# Ret2Func

gets(), strcpy() = Shell

Win Addr = 0x403400

RBP

RBP - 0x20 = RSP

## Prog. Stack

AAAAAAAA

AAAAAAAA

Return Addr AAAAAAAA

SPO AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

Buffer [0x20] AAAAAAAA

# Ret2Func

gets(), strcpy() = Shell

Win Addr = 0x403400

코드 흐름을 탈취

RBP

$\text{RBP} - 0x20 = \text{RSP}$

## Prog. Stack

AAAAAAAA

AAAAAAAA

Return Addr 00403400

SPO AAAAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

Buffer [0x20] AAAAAAAAAA

# Ret2ShellCode

+ use Shellcode

If Win 함수가 없다

스택에 명령어(셸)를 올리자

이후 그 스택 위치로 return

RBP

$\text{RBP} - 0x20 = \text{RSP}$

## Prog. Stack

AAAAAAAA

AAAAAAAA

Return Addr 00403400

SPO AAAAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

Buffer [0x20] AAAAAAAAAA

# Ret2ShellCode

+ use Shellcode

shell code 위치로 ret

RBP

$RBP - 0x20 = RSP$

## Prog. Stack

AAAAAAAA

AAAAAAAA

Return Addr

Stk Addr

SPO

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

AAAAAAAA

SomeCode

SomeCode

Buffer [0x20]

SomeCode



# 실습 [Ret2Shellcode]

notion에 업로드 된 Ret2Shellcode.zip

2025.09.17

# Q&A

질문이 있다면 하십시오

Guardian 2025

2.5cm-2.5cm 떨어진 제목 36px

제목 하단의 부제목 18px

3.5cm 떨어진 내용 1 32px

좌측으로 0.5cm 떨어진 내용 하단의 설명 18px

3.5cm 떨어진 내용 2 32px

좌측으로 0.5cm 떨어진 내용 하단의 설명 18px

3.5cm 떨어진 내용 3 32px

좌측으로 0.5cm 떨어진 내용 하단의 설명 18px

1cm-1cm 떨어진 주석 12px

1cm-1cm 떨어진 주석 12px