

2025.09.24

ROP

가디언 시스템 보안 & 취약점 분석 세미나 2.2

임준서

Security Mitigations

ASLR, PIE, NX, RELRO, CANARY

I

Security Mitigation

Summary

ASLR: stack 아무데나

Always on. Address Space Layout Randomization

NX: stack에서 shellcode 금지

No eXecute

PIE: .text 아무데나

Position Independent Executables

Security Mitigation

Summary

RELRO: .got overwrite 금지

RELocation Read-Only. 그리고 정확히는 Full RELRO

Canary: 스택에 parity 추가

Therefore, BOF 하다가 parity 건드리면 프로그램 터짐

Windows, Stack Cookie. Linux, Stack Canary

[카나리아] 또는 [카나리]라고 읽는다

Security Mitigation

ASLR

Address Space Layout Randomization

Security Mitigation

NX

No-eXecute

Security Mitigation

PIE

Position Independent Executables

Security Mitigation

RELRO

RELocation Read-Only

Security Mitigation

Canary

Stack Smashing Protection

Security Mitigation

Canary

stack에 검증 값 삽입

return 과정에서 확인

틀리면 터짐

stack smashing detected!

첫번째 값은 \x00으로 고정. (출력 방지용)

Security Mitigation

NX

Ret2ShellCode는 NX 옵션에서 불가

대부분 NX on

ROP

Ret2_____

] [

Return Oriented Programming

a.k.a. ROP

Somewhere in the code ...

(0x00400700)
pop rdi;
ret;

(0x00400900)
call system;
ret;

Ret2ShellCode

pop 하고 ret 하는 코드 조각을 사용하자!

RBP

RBP - 0x20 = RSP

Prog. Stack	
	Call sys
	bin/sh
	pop %rdi
Return Addr	retq
SPO	AAAAAAA
	AAAAAAA
Buffer [0x20]	AAAAAAA

ROP

ReturnOrientedProgramming

pop __; ret;
코드 조각들을 모아 조작

RBP

RBP - 0x20 = RSP

Prog. Stack

	00400900
	/sh addr
Return Addr	00400700
SPO	AAAAAAA
	AAAAAAA
Buffer [0x20]	AAAAAAA

ROP

Return Oriented Programming

ROP에 쓰이는 코드조각을
gadget이라 한다.

RBP

RBP - 0x20 = RSP

Prog. Stack

	00400900
	/sh addr
Return Addr	00400700
SPO	AAAAAAA
	AAAAAAA
Buffer [0x20]	AAAAAAA

도구라는 뜻임

ROP

ReturnOrientedProgramming

call read

8

pop rdx; ret;

addr of GOT

pop rsi; ret

0

pop rdi;ret;

RBP

RBP - 0x20 = RSP

Prog. Stack

...	pop %rdi
Return Addr	retq
SPO	AAAAAAA
	AAAAAAA
Buffer [0x20]	AAAAAAA

ROP

ReturnOrientedProgramming

call read

8

pop rdx; ret;

addr of GOT

pop rsi; ret

0

pop rdi;ret;

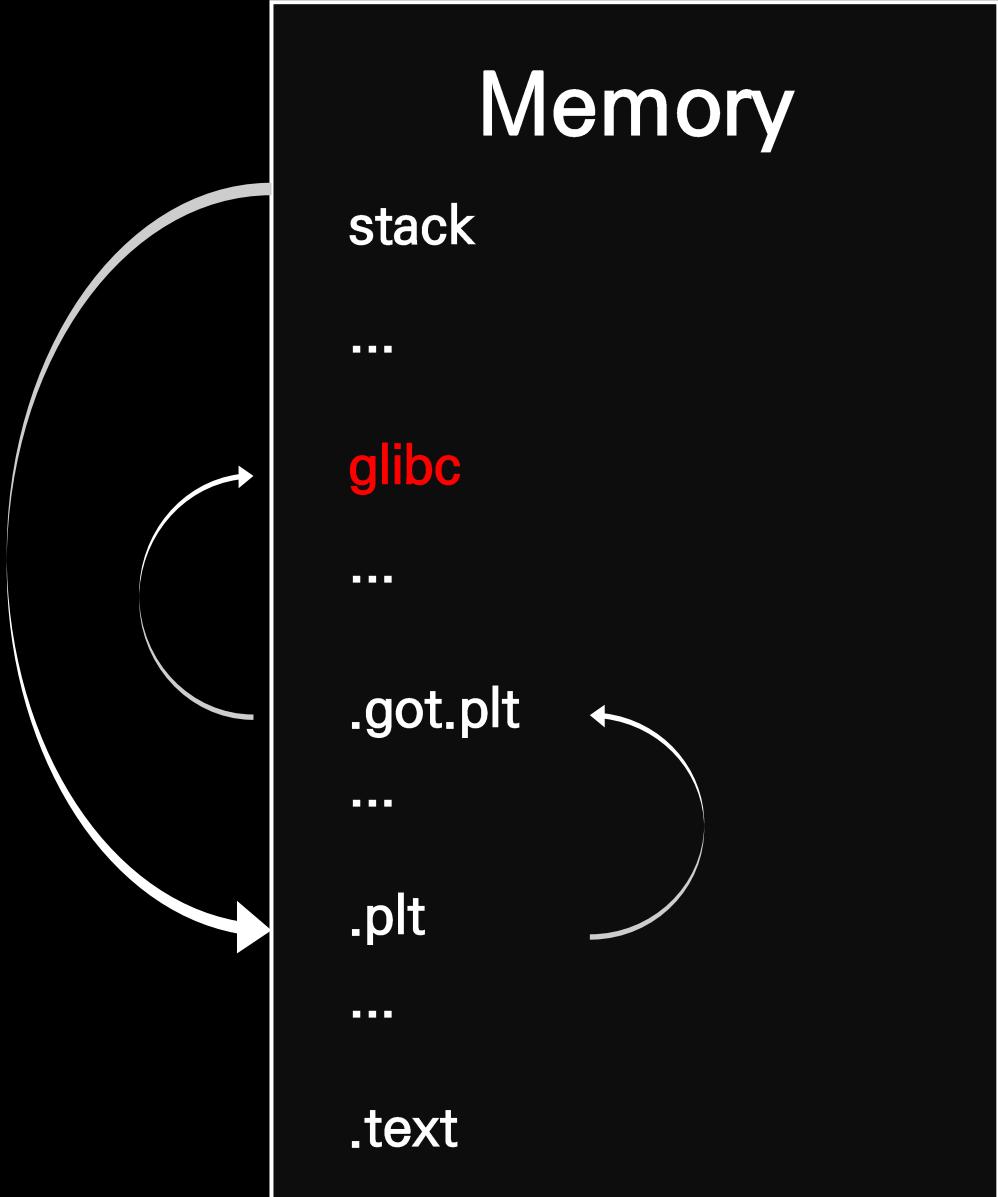
read(0, GOT, 8);

RB_P - 0x20 = RS_P

Prog. Stack

	...
	pop %rdi
Return Addr	retq
SPO	AAAAAAA
	AAAAAAA
Buffer [0x20]	AAAAAAA

Library



glibc의 위치는 매번 바뀜 (ASLR...)

단, 하위 12비트(3글자는 고정.)

Library

```
.plt:000000000400620 ; int puts(const char *s)
=plt:000000000400620 _puts          proc near             ; CODE XREF: main+41↓p
=plt:000000000400620                 jmp    cs:off_601018
=plt:000000000400620 _puts          endp
=plt:000000000400620
=plt:000000000400626 ;
=plt:000000000400626         push   0
=plt:00000000040062B         jmp    sub_400610
=plt:000000000400630
=plt:000000000400630 ; ===== S U B R O U T I N E =====
=plt:000000000400630
=plt:000000000400630 ; Attributes: thunk
=plt:000000000400630
=plt:000000000400630 ; size_t fread(void *ptr, size_t size, size_t n, FILE *stream)
=plt:000000000400630 _fread         proc near             ; CODE XREF: main+C1↓p
=plt:000000000400630                 jmp    cs:off_601020
=plt:000000000400630 _fread         endp
=plt:000000000400630
=plt:000000000400636 ;
=plt:000000000400636         push   1
=plt:00000000040063B         jmp    sub_400610
=plt:000000000400640
=plt:000000000400640 ; ===== S U B R O U T I N E =====
=plt:000000000400640
=plt:000000000400640 ; Attributes: thunk
=plt:000000000400640
=plt:000000000400640 ; int fclose(FILE *stream)
=plt:000000000400640 _fclose        proc near             ; CODE XREF: main+D0↓p
=plt:000000000400640                 jmp    cs:off_601028
=plt:000000000400640 _fclose        endp
=plt:000000000400646 :
```

Library

```
.got.plt:0000000000601000 ; =====
.got.plt:0000000000601000
.got.plt:0000000000601000 ; Segment type: Pure data
.got.plt:0000000000601000 ; Segment permissions: Read/Write
.got.plt:0000000000601000 _got_plt      segment qword public 'DATA' use64
.got.plt:0000000000601000           assume cs:_got_plt
.got.plt:0000000000601000           ;org 601000h
.got.plt:0000000000601000 _GLOBAL_OFFSET_TABLE_ dq offset _DYNAMIC
.got.plt:0000000000601008 qword_601008    dq 0                      ; DATA XREF: sub_400610↑r
.got.plt:0000000000601010 qword_601010    dq 0                      ; DATA XREF: sub_400610+6↑r
.got.plt:0000000000601018 off_601018     dq offset puts          ; DATA XREF: _puts↑r
.got.plt:0000000000601020 off_601020     dq offset fread          ; DATA XREF: _fread↑r
.got.plt:0000000000601028 off_601028     dq offset fclose         ; DATA XREF: _fclose↑r
.got.plt:0000000000601030 off_601030     dq offset __stack_chk_fail
                                         ; DATA XREF: __stack_chk_fail↑r
.got.plt:0000000000601038 off_601038     dq offset printf         ; DATA XREF: _printf↑r
.got.plt:0000000000601040 off_601040     dq offset __libc_start_main
                                         ; DATA XREF: __libc_start_main↑r
.got.plt:0000000000601040
.got.plt:0000000000601048 off_601048     dq offset ftell          ; DATA XREF: _ftell↑r
.got.plt:0000000000601050 off_601050     dq offset fseek          ; DATA XREF: _fseek↑r
.got.plt:0000000000601058 off_601058     dq offset fopen          ; DATA XREF: _fopen↑r
.got.plt:0000000000601060 off_601060     dq offset fwrite         ; DATA XREF: _fwrite↑r
.got.plt:0000000000601060 _got_plt      ends
```

ROP

GOT Overwrite

Read 호출 → PLT 호출

Procedure Linkage Table

read의 GOT 확인

system으로 overwrite

Read 실행 = system 호출

잘못 조작하면 프로그램 터짐

heap 조작으로 AAR/AAW할 때도 종종 쓰임

Security Mitigation

RELRO

RELocation Read-Only

No RELRO

PARTIAL RELRO

FULL RELRO

실습 [ROP]

dreamhack.io/wargame/challenges/354

```
$ checksec rop[*]
' /home/dreamhack/rop'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

```
int main() {
    char buf[0x30];

    setvbuf(stdin, 0, _IONBF, 0);
    setvbuf(stdout, 0, _IONBF, 0);

    // Leak canary
    puts("[1] Leak Canary");
    write(1, "Buf: ", 5);
    read(0, buf, 0x100);
    printf("Buf: %s\n", buf);

    // Do ROP
    puts("[2] Input ROP payload");
    write(1, "Buf: ", 5);
    read(0, buf, 0x100);

    return 0;
}
```

Homework [dowell]

dreamhack.io/wargame/challenges/1568

여러번 입력을 받을 수는 없을까?

2025.09.24

Q&A

질문이 있다면 하십시오

임준서

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

2.5cm-3.5cm 떨어진 제목 36px

제목 하단의 부제목 18px

3.5cm 떨어진 내용 1 32px

Git init

Git status

Git add text.txt

Git add .

Git commit

Ctrl+C

Git commit -m “genesis”

Git log

Git log --oneline

Git add .

Git reset .

Git commit -m “add README”

Git log --oneline -n 3

Git commit -a -m “hello”

1cm-1cm 떨어진 주석 12px

1cm-1cm 떨어진 주석 12px

중심에서 0.3cm 떨어진 소속 18px

ROP

Ret2main

MOVED TO 3_1

기회가 1번이면 ROP 어려움

return to main!

여러번 ROP가 가능하다

ROP 中 RTL MOVED TO 3_1

Return To Libc

원하는 gadget이 없다면?

Return To Libc!

libc에 좋은 gadget이 차고 넘친다. system, /bin/sh 까지 다 있다.

굉장히 유명하고 간편함

대신 libc의 시작 주소를 알아야 함.

ROP 中 RTL MOVED TO 3_1

One_gadget

libc 에 좋은게 많으니 한번에 쉘을 따자

onegadget ./libc.so.6

조건만 맞으면 한번에 쉘이 나온다.

libc의 버전이 높을수록 onegadget의 제약 조건이 많아진다