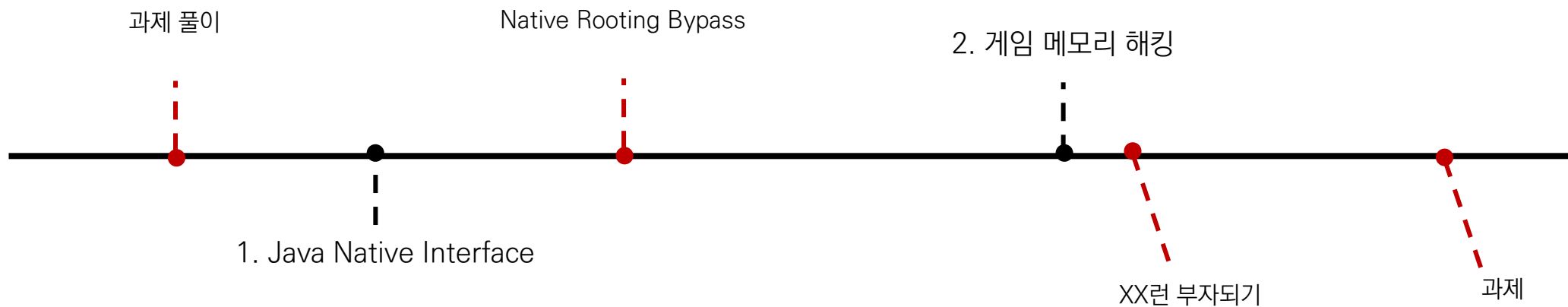


2020 안드로이드 앱 해킹 교육

PART 3. Java Native Interface & Memory Cheating

금일 교육 강의 구성

강의 타임라인



주의 사항

환경 세팅이 안되었을 때 확인해볼 사항

- 환경 변수 설정시 ADB가 아닌 녹스 설치 경로로 잘 설정 되어있는가?
- 환경변수 설정 후 환경 변수 창을 “확인“ 버튼을 통해 종료하고 cmd를 꺾다 켜는가?
- frida는 rooting이 된 상태에서에서만 동작
- 안될 경우 오류 메시지 구글에 검색

Frida lab 6

6. Run chall06() after 10 seconds with correct value

“10초 뒤에“ → setTimeout(~~~, 10000);

“올바른 인자로”

“chall06 호출”

Frida lab 6

6. Run chall06() after 10 seconds with correct value

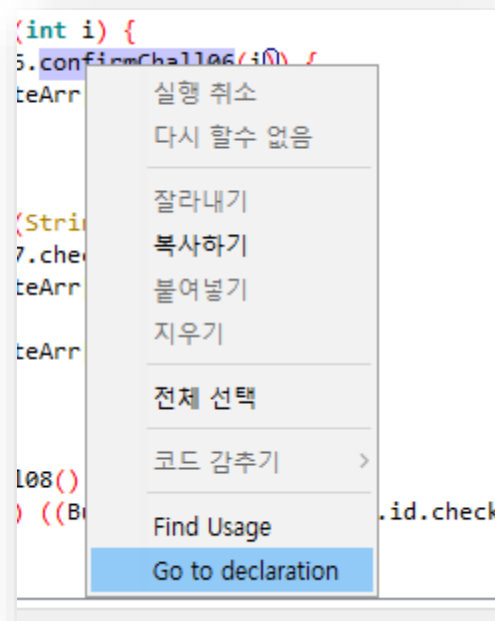
“10초 뒤에“  setTimeout(~~~, 10000);

“올바른 인자로”

“chall06 호출”

Frida lab 6

```
public void chall06(int i) {  
    if (challenge_06.confirmChall06(i)) {  
        this.completeArr[5] = 1;  
    }  
}
```



```
public static boolean confirmChall06(int i) {  
    return i == chall06 && System.currentTimeMillis() > timeStart + 10000;  
}  
  
public static void addChall06(int i) {  
    chall06 += i;  
    if (chall06 > 9000) {  
        chall06 = i;  
    }  
}
```

○ ㄹ i값이랑 chall06이랑 똑같아야하고 시작시간으로 부터(아마도 어플 시작 시간일것) 10초 “이상“ 지나야한데

근데 addchall06은 무엇? 우리가 구하고자 하는 chall06에 값이 계속 더해짐

Frida lab 6

Node	Code
uk.rossmarks.fridalab.MainActivity.onCreate(Bundle) void	challenge_06.addChall06(nextInt);
uk.rossmarks.fridalab.MainActivity.onCreate(Bundle) void	challenge_06.addChall06(new Random().nextInt(50) + 1);
uk.rossmarks.fridalab.challenge_06.addChall06(int) void	public static void addChall06(int i) {

```
challenge_06.startTime();
challenge_06.addChall06(new Random().nextInt(50) + 1);
new Timer().scheduleAtFixedRate(new TimerTask() {
    public void run() {
        int nextInt = new Random().nextInt(50) + 1;
        challenge_06.addChall06(nextInt);
        Integer.toString(nextInt);
    }
}, 0, 1000);
```

onCreate -> 화면이 생성 되었을때

즉, 어플이 시작된 시점부터 1초에 한번씩 chall06(우리가 구해야 하는 값)이 바뀌는데 그게 랜덤..임

과제 풀이

Frida lab 6

“10초 뒤에“

“올바른 인자로”

“chall06 호출”  chall06(chall06); 하란 소리

그럼 그냥 “객체 변수를 후킹 ” 해서 10초뒤에 chall06 인자로 넣으면 끝?

Frida lab 6

1, 2, .. 10, 11 초
즉, 매초마다 호출됨

```
setTimeout(function(){ // after 10 sec
  setImmediate(function(){ // prevent timeout
    Java.perform(function(){ // when JVM loaded
      var _class = Java.use("uk.rossmarks.fridalab.challenge_06"); // target class
      _class.addChall06.implementation = function(arg){
        Java.choose("uk.rossmarks.fridalab.MainActivity", {
          onMatch : function(instance) {
            var variable_i = _class.chall06.value // variable "i"
            instance.chall06(variable_i); // solve
          },
          onComplete : function(){
            console.log("[*] 10 sec later.."); // log
          }
        })
      })
    })
  },10000); // 10000 밀리초 == 10초
```

chall06(chall06) 호출

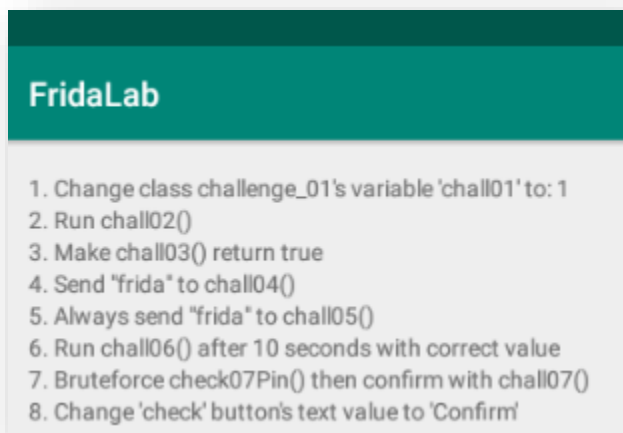
과제 풀이

Frida lab 6

1. Change class challenge_01's variable 'chall01' to: 1
2. Run chall02()
3. Make chall03() return true
4. Send 'frida' to chall04()
5. Always send 'frida' to chall05()
6. Run chall06() after 10 seconds with correct value
7. Bruteforce check07Pin() then confirm with chall07()
8. Change 'check' button's text value to 'Confirm'

과제 풀이

Frida lab 7



“브루트포스해라”
“맞으면 chall07에 넣어라 ”

Frida lab 7

```
public void chall07(String str) {  
    if (challenge_07.check07Pin(str)) {  
        this.completeArr[6] = 1;  
    } else {  
        this.completeArr[6] = 0;  
    }  
}
```

```
public class challenge_07 {  
    static String chall07;  
  
    public static void setChall07() {  
        chall07 = BuildConfig.FLAVOR + (((int) (Math.random() * 9000.0d)) + 1000);  
    }  
  
    public static boolean check07Pin(String str) {  
        return str.equals(chall07);  
    }  
}
```

check07Pin() -> 핀이 맞는가? 즉, 여기에 1000부터 9999까지 대입했을 때 true가 나오는 값이 맞는 핀
-> 그 핀을 가져다가 MainActivity.chall07에 전달

Frida lab 7

```
Java.perform(function(){
    var main = null;

    Java.choose('uk.rossmarks.fridalab.MainActivity', {
        onMatch: function(args) {    main = args;    },
        onComplete: function() {}
    });

    var _class = Java.use('uk.rossmarks.fridalab.challenge_07')
    for(let i = 1000; i < 10000; i++) {
        let isPinTrue = _class.check07Pin(i.toString());
        if(isPinTrue) {
            main.chall07(i.toString())
            break;
        }
    }
});
```

과제 풀이

Frida lab 7

FridaLab

1. Change class challenge_01's variable 'chall01' to: 1
2. Run chall02()
3. Make chall03() return true
4. Send "frida" to chall04()
5. Always send "frida" to chall05()
6. Run chall06() after 10 seconds with correct value
7. Bruteforce check07Pin() then confirm with chall07()
8. Change 'check' button's text value to 'Confirm'

CHECK

Java Native Interface

- JVM(Java Virtual Machine)에서 동작하는 코드는 안정적이며 호환성이 좋다.
- 하지만 가상의 머신 위에서 돌아가기 때문에 운영체제에서 지원하는 모든 기능(Native)을 사용할 수는 없다
- 때문에 안드로이드에서는 Java 코드에서 C, C++, 어셈블리로 이루어진 함수를 부를 수 있도록 JNI를 제공한다.

동적 라이브러리 모듈

Java Native Interface

+ 추가) JNI를 이용하여 루팅을 탐지하는 경우도 있다.

fopen, access, ..

동적 라이브러리 모듈

Java Native Interface

System.loadLibrary() → APK 내부에서 동적 라이브러리(*.so) 탐색 후 JNI_onLoad() → 메모리에 적재 → Java 코드 동작

```
static {  
    System.loadLibrary("bpsec");  
}
```

```
this.rootApi.isRooting()
```

libbpsec.so가 APK에 내장되어 있음을 알 수 있다.

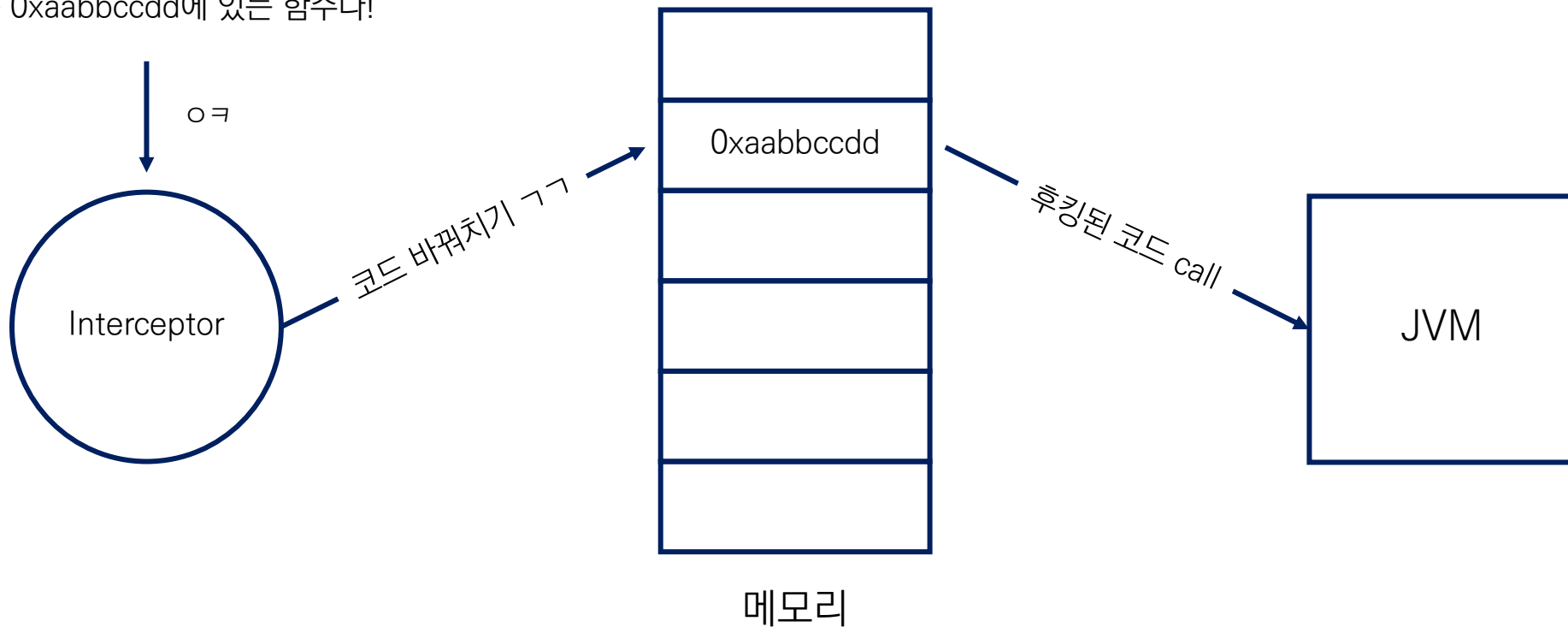
Java Native Interface

어떻게 후킹할까? Interceptor

`Interceptor.attach(target, callbacks[, data])` : intercept calls to function at `target`. This is a `NativePointer` specifying the address of the function you would like to intercept calls to. Note that on 32-bit ARM this address must have its least significant bit set to 0 for ARM functions, and 1 for Thumb functions. Frida takes care of this detail for you if you get the address from a Frida API (for example `Module.getExportByName()`). The `callbacks` argument is an object containing one or more of:

Java Native Interface

타겟은 0xaabbccdd에 있는 함수다!



Java Native Interface

우리가 원하는 함수 주소는 어떻게 아느냐?

```
Module.findExportByName("lib~~.so", "printf")
```




Native Pointer를 return해줌

Java Native Interface

```
Interceptor.attach(Native Pointer, {  
    onEnter: function(args) { },  
    onLeave: function(retval) { retval.replace(-1); }  
});
```

Java Native Interface

Native Pointer : 메모리 상에 위치

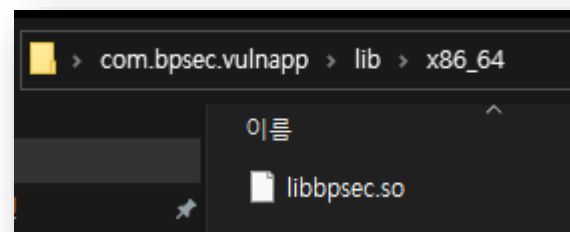
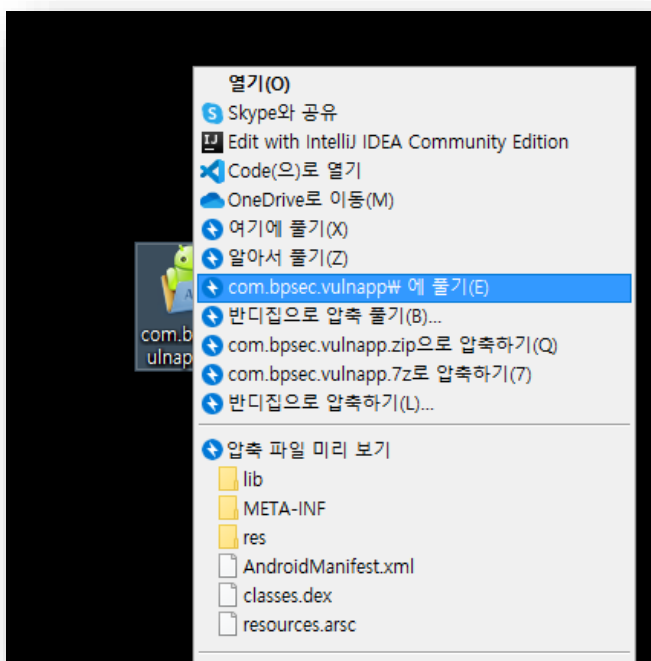


```
Interceptor.attach(Module.findExportByName("lib~~", "printf"), {  
    onEnter: function(args) { },  
    onLeave: function(retval) { retval.replace(-1); }  
});
```

Java Native Interface

(* .so) 추출

기본적으로 APK는 어떠한 구조를 지니고 있는 “압축 파일” 형태임 == 압축 해제시 구조 파악 가능



lib > x86_64

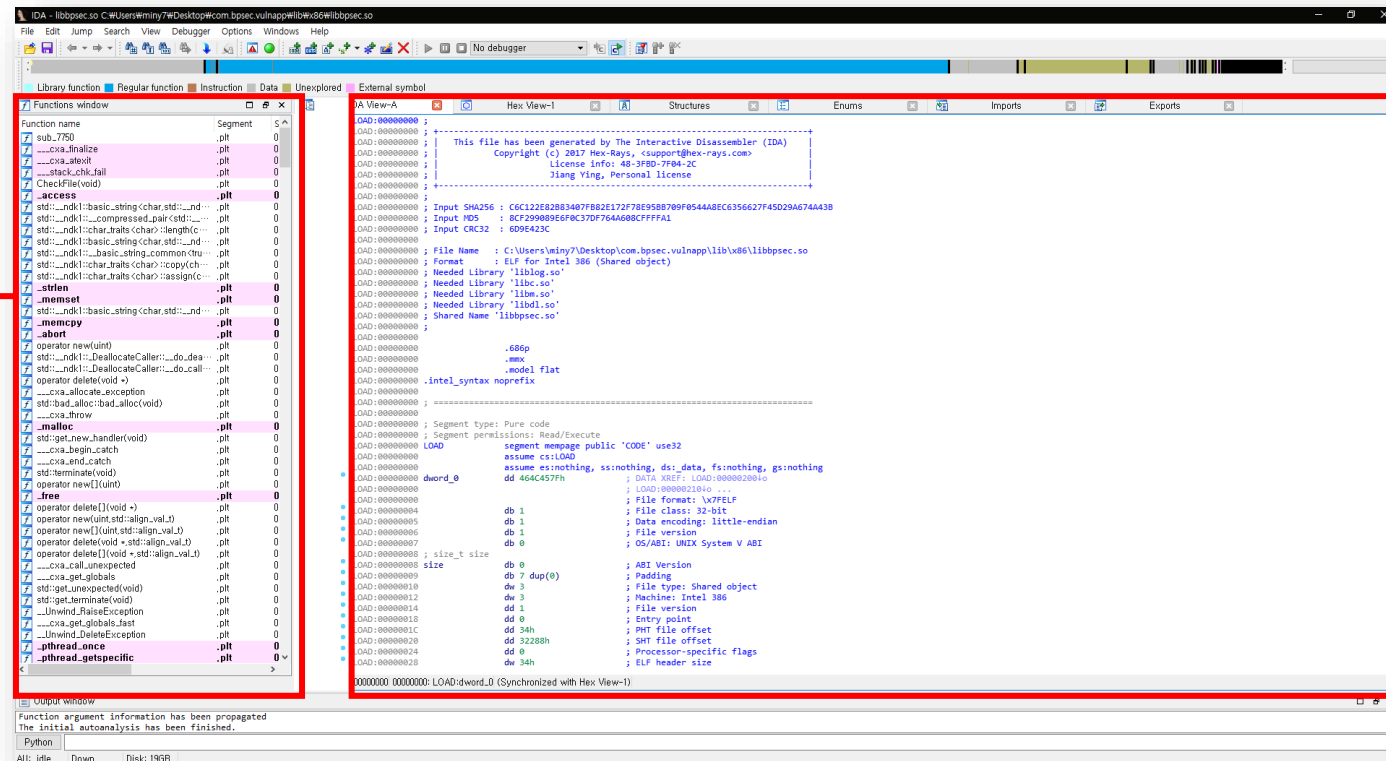
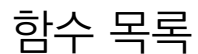
동적 라이브러리 모듈

Java N

동적 라이브러리 모듈

Java Native Interface

추출한 (*.so) 분석 -> ida
ida는 assembly to c가 가능한 decompiler이다.



함수를 선택하면
어셈블리를 보여줌

Java Native Interface

```
char *v15; // [esp+10h] [ebp-1Ch]
char *v16; // [esp+10h] [ebp-1Ch]
char **v17; // [esp+14h] [ebp-18h]
char *v18; // [esp+18h] [ebp-14h]

v3 = *v2;
v4 = v2[1];
if ( *v2 == v4 )
{
    v5 = *v2;
}
else
{
    v5 = *v2;
    if ( *v3 == 110 )
    {
        v5 = v3 + 1;
        *v2 = v3 + 1;
    }
}
result = 0;
v17 = v2;
if ( v5 == v4 || (v15 = v3, (unsigned int)(*v5 - 48) > 9) )
{
    v9 = 0;
    v10 = 0;
}
else
{
    .
}
```

F5를 누르면 C코드로 변환 -> 사실상 ida를 쓰는 가장 큰 이유

동적 라이브러리 모듈

Java Native Interface

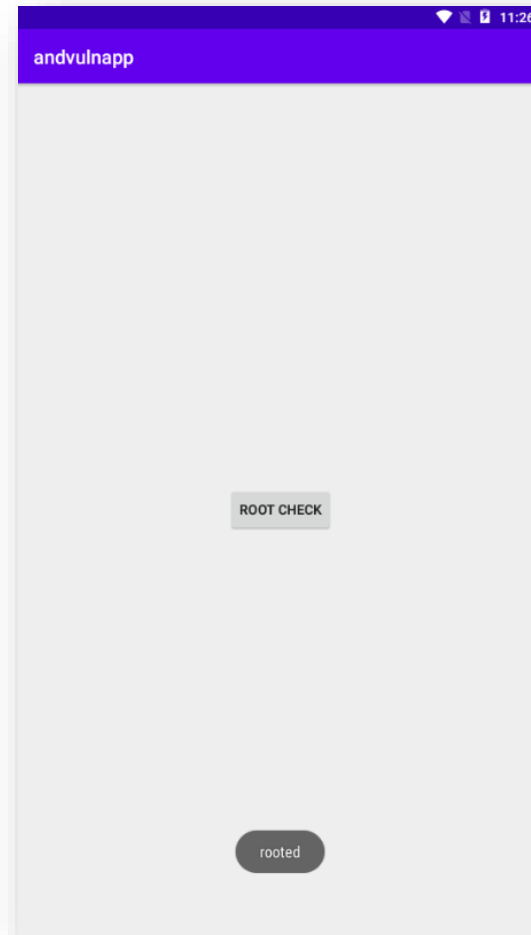
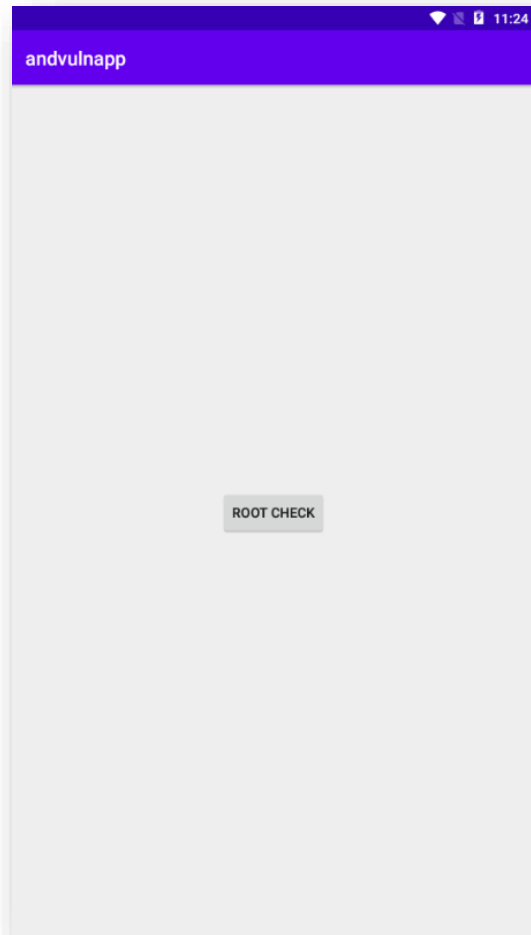
분석 프로세스

jadx로 dex decompile 후 loadLibrary 체크 -> 라이브러리 명(*.so) 캐치 후 APK Unpack -> 해당 lib추출하여 ida에 분석

-> 충분한 소스 분석 후 후킹할 함수 선정 -> frida의 Interceptor로 후킹

실습 1 – Native Rooting Bypass

<http://bpsec.co.kr/~minibeef/download/com.bpsec.vulnapp.apk>



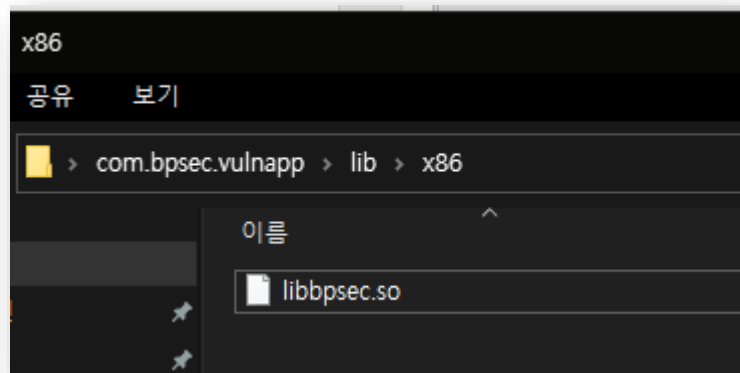
실습 1 – Native Rooting Bypass

```
public class MainActivity extends AppCompatActivity {  
    RootChecker rootApi = new RootChecker();  
  
    /* access modifiers changed from: protected */  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView((int) R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.rootcheck);  
    }  
  
    public void rootcheckClick(View view) {  
        Toast.makeText(this, this.rootApi.isRooting() ? "rooted" : "unrooted", 1).show();  
    }  
}
```

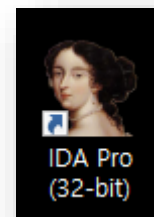
```
public class RootChecker {  
    public native boolean isRooting();  
  
    static {  
        System.loadLibrary("bpsec");  
    }  
}
```

loadLibrary 호출 확인 → libbpsec.so에서 루팅 탐지 루틴이 있는 것으로 보임

실습 1 – Native Rooting Bypass



so 파일 추출 후(x86으로 해야함)



IDA에 Drag&Drop

실습 1 – Native Rooting Bypass

```

LOAD:000... 00000008 C 5594570
LOAD:000... 00000000 C __cxa_atexit
LOAD:000... 00000005 C LIBC
LOAD:000... 00000008 C libc.so
LOAD:000... 0000000C C libbsec.so
LOAD:000... 0000000F C __cxa_finalize
LOAD:000... 00000011 C __stack_chk_fail
LOAD:000... 00000010 C dl_iterate_phdr
LOAD:000... 00000009 C libdl.so
LOAD:000... 0000002E C Java_com_bpsec_libbsec_RootChecker_IsRooting
LOAD:000... 0000000E C _Z9CheckFilev
LOAD:000... 00000043 C _ZNKSt6__ndk121__basic_string_commonLb1EE20__throw_length_erro...
LOAD:000... 00000029 C _ZNSt6__ndk111char_traitsLcE4copyEPcPKc
LOAD:000... 0000000F C pthread_create
LOAD:000... 0000002A C _ZNSt6__ndk111char_traitsLcE6assignERcRKc
LOAD:000... 00000028 C _ZNSt6__ndk111char_traitsLcE6lengthEPKc
LOAD:000... 00000059 C _ZNSt6__ndk112basic_stringLcNS_11char_traitsLcEENS_9allocatorEE...
LOAD:000... 00000009 C memalign
LOAD:000... 0000004F C _ZNSt6__ndk112basic_stringLcNS_11char_traitsLcEENS_9allocatorEE...
LOAD:000... 00000047 C _ZNSt6__ndk112basic_stringLcNS_11char_traitsLcEENS_9allocatorEE...
LOAD:000... 00000008 C __errno
LOAD:000... 00000041 C _ZNSt6__ndk117_DeallocateCaller27__do_deallocate_handle_sizeEPvj
LOAD:000... 0000002D C _ZNSt6__ndk117_DeallocateCaller9__do_callEPv
LOAD:000... 00000070 C _ZNSt6__ndk117__compressed_pairINS_12basic_stringLcNS_11char_tra...
LOAD:000... 00000007 C _ZdlPv
LOAD:000... 00000007 C strcmp
LOAD:000... 00000006 C _Znwj
LOAD:000... 00000012 C __stack_chk_guard
LOAD:000... 00000006 C abort
LOAD:000... 00000007 C access
LOAD:000... 00000007 C memcpy
LOAD:000... 00000007 C memset
LOAD:000... 00000007 C strlen
LOAD:000... 00000014 C _ZNSt9bad_allocC1Ev
LOAD:000... 00000014 C _ZNSt9bad_allocD1Ev
LOAD:000... 00000017 C _ZSt15get_new_handlerv
LOAD:000... 00000019 C _ZSt17__throw_bad_allocv
LOAD:000... 0000000D C _ZSt7nothrow

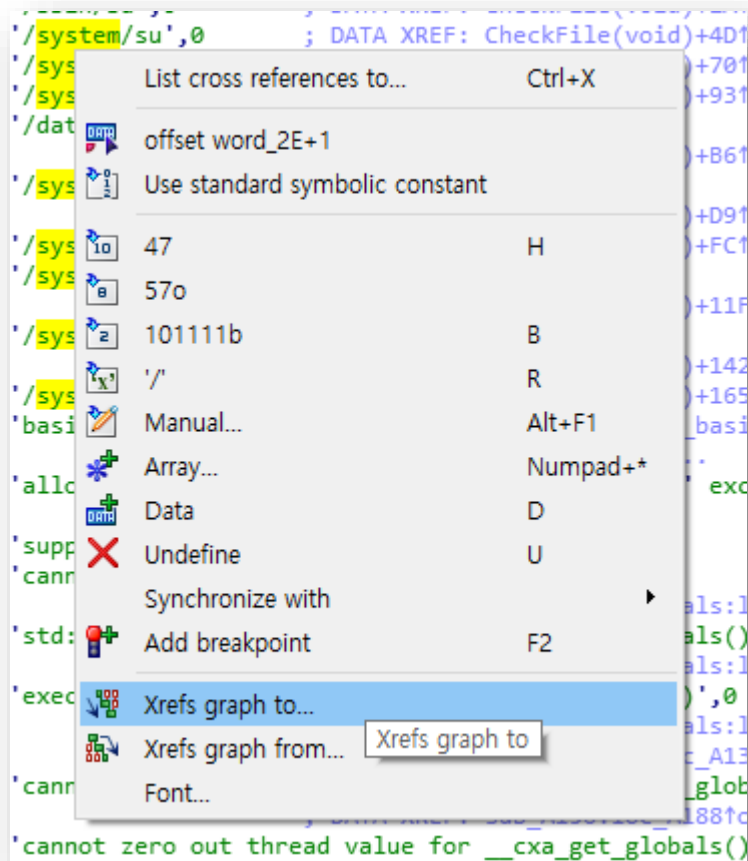
```

Address	Length	Type	String
.rodata:0...	0000000B	C	/system/su
.rodata:0...	00000010	C	/system/sbin/su
.rodata:0...	00000010	C	/system/xbin/su
.rodata:0...	0000001A	C	/system/app/Superuser.apk
.rodata:0...	0000000F	C	/system/bin/su
.rodata:0...	00000015	C	/system/bin/.ext/.su
.rodata:0...	00000023	C	/system/usr/we-need-root/su-backup
.rodata:0...	00000010	C	/system/xbin/mu

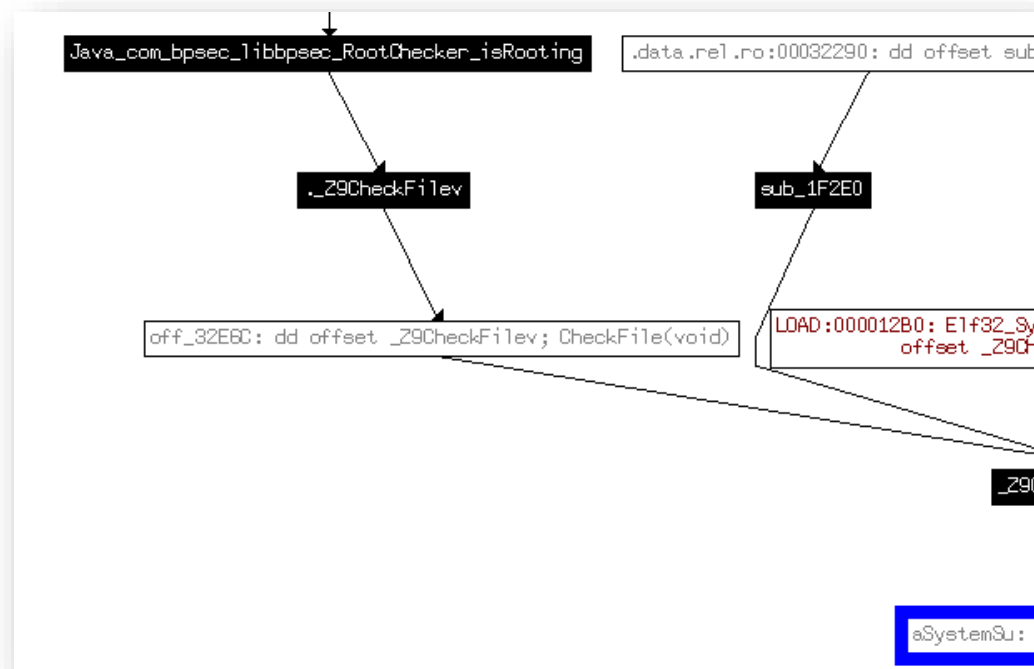
Ctrl + F로 루팅 파일 검색

Shift + F12 누르면 문자열 리스트 출력

실습 1 – Native Rooting Bypass

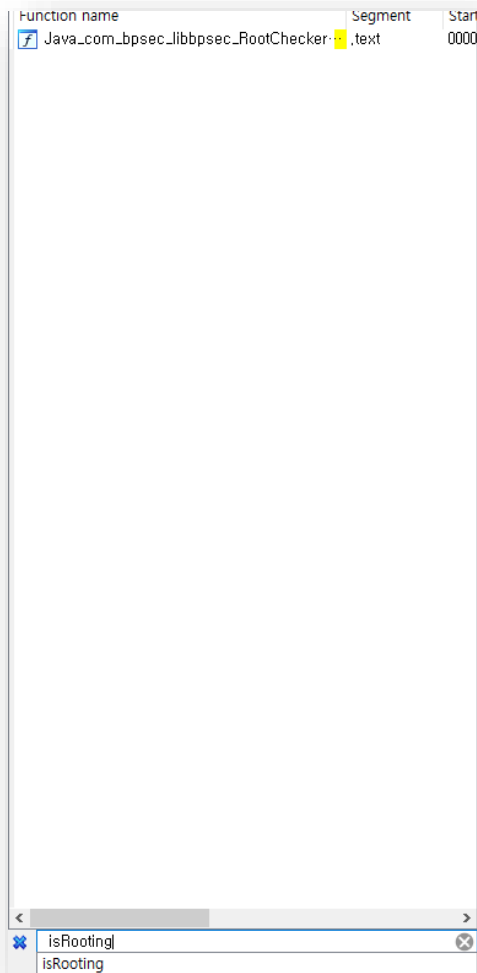


문자열 더블클릭 -> 이동한 페이지에서 마우스 오른쪽 -> Xrefs graph to
이 문자열을 참조하는 위치를 알려줌



isRooting이라는 함수가 /system/bin/su를 참조하는 것을 확인

실습 1 – Native Rooting Bypass



```
sub_8160(s, (int)"/sbin/su");
sub_8160(v6, (int)"/system/su");
sub_8160(_68, (int)"/system/sbin/su");
sub_8160(_74, (int)"/system/sbin/su");
sub_8160(_80, (int)"/data/data/com.noshufou.android.su");
sub_8160(_8C, (int)"/system/app/Superuser.apk");
sub_8160(_98, (int)"/system/bin/su");
sub_8160(_A4, (int)"/system/bin/.ext/.su");
sub_8160(_B0, (int)"/system/usr/we-need-root/su-backup");
sub_8160(_BC, (int)"/system/sbin/mu");
for ( i = 0; i < 10; ++i )
{
    v0 = (const char *)sub_81E0(&s[12 * i]);
    if ( !access(v0, 0) )
    {
        v4 = 1;
        goto LABEL_7;
    }
}
v4 = 0;
LABEL_7:
v2 = _C8;
do
```

access는 파일을 여는 함수임

좌측 함수 목록에서 isRooting 검색 -> 더블클릭

실습 1 – Native Rooting Bypass

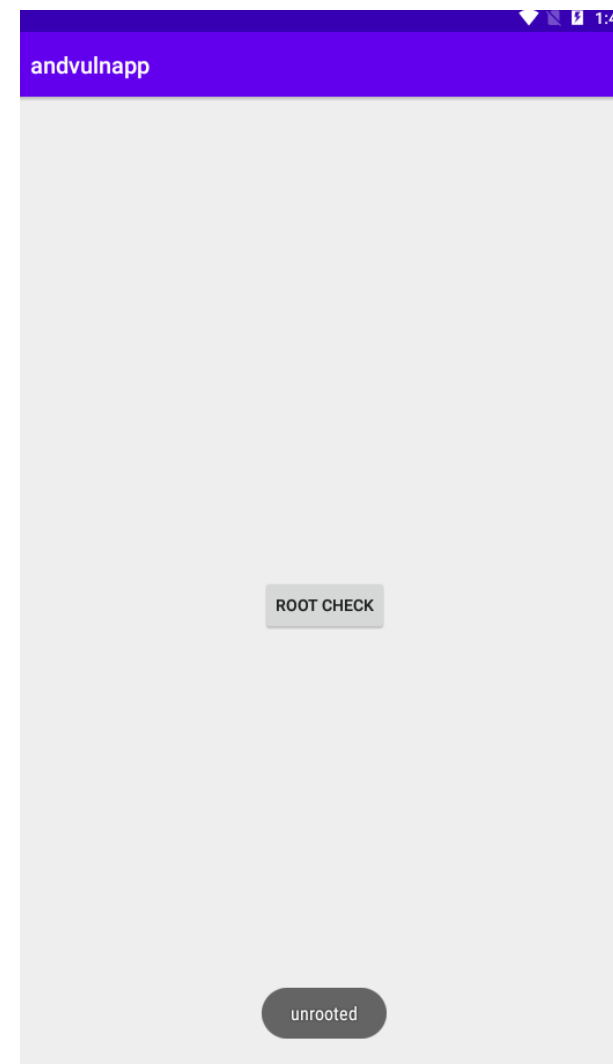
Return Value

파일에 지정된 모드가 있으면 각 함수는 0을 반환합니다. 명명 된 파일이 없거나 지정 된 모드가 없는 경우 함수는 -1을 반환 합니다. 이 경우 `errno` 는 다음 표와 같이 설정 됩니다.

실습 1 – Native Rooting Bypass

```
Java.perform(function() {  
    Interceptor.attach(Module.findExportByName("libbpsec.so", "access"), {  
        onEnter: function(args) { console.log("[*] access() Called"); },  
        onLeave: function(retval) { retval.replace(-1); console.log("[*] access returns " + retval); }  
    });  
});
```

```
[*] access returns 0xffffffff  
[*] access() Called  
[*] access returns 0xffffffff  
[*] access() Called  
[*] access returns 0xffffffff  
[*] access() Called  
[*] access returns 0xffffffff  
[*] access() Called  
[*] access returns 0xffffffff  
[*] access() Called  
[*] access returns 0xffffffff
```



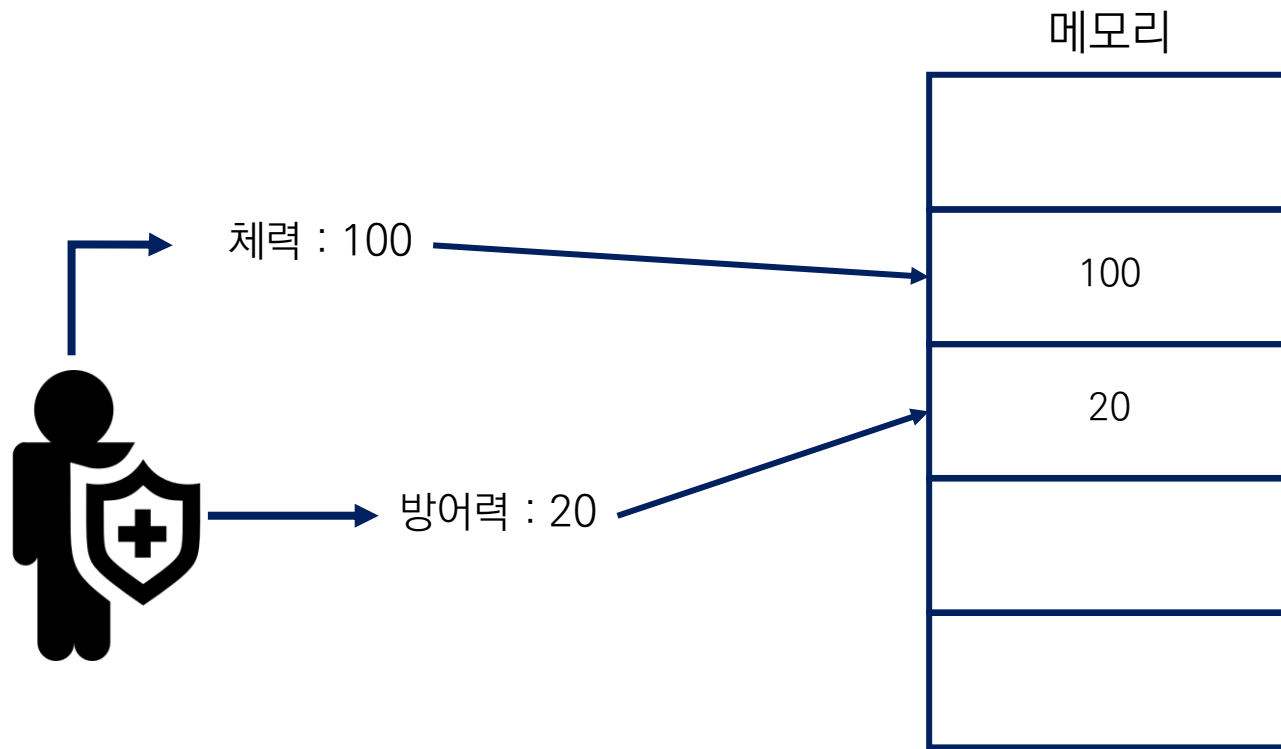
Memory Cheating

게임 메모리 해킹



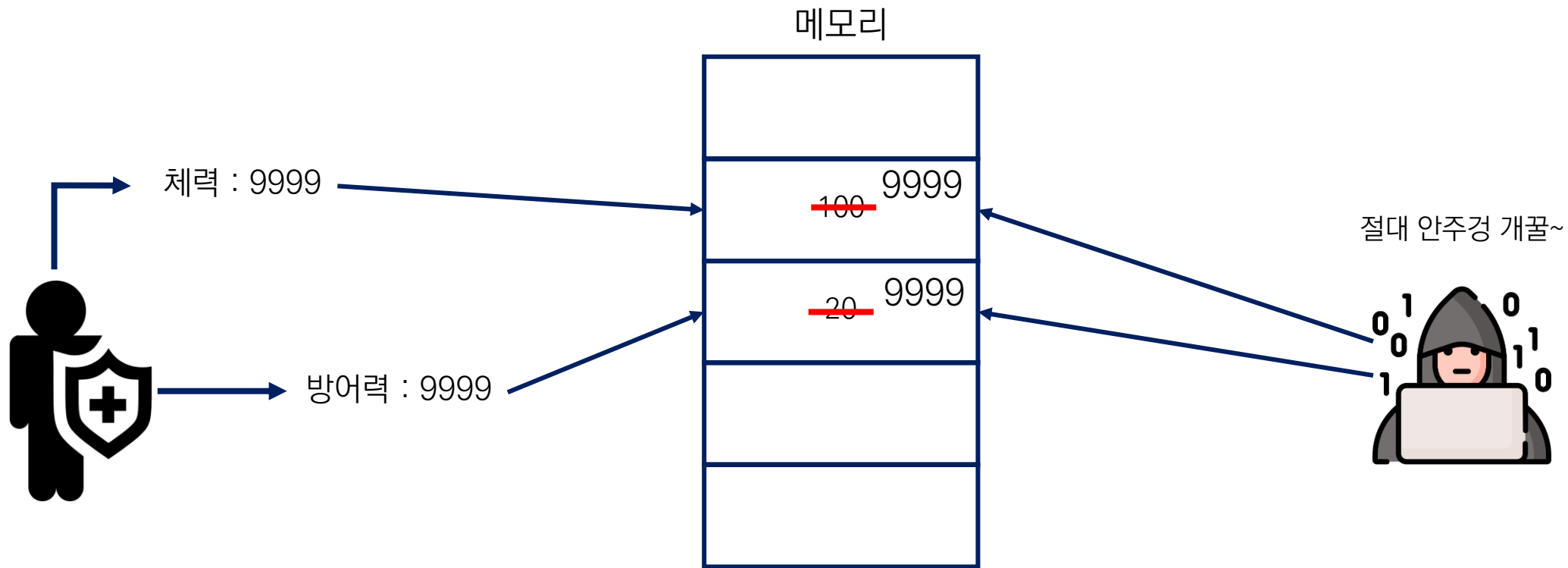
이런거..

게임 메모리 해킹



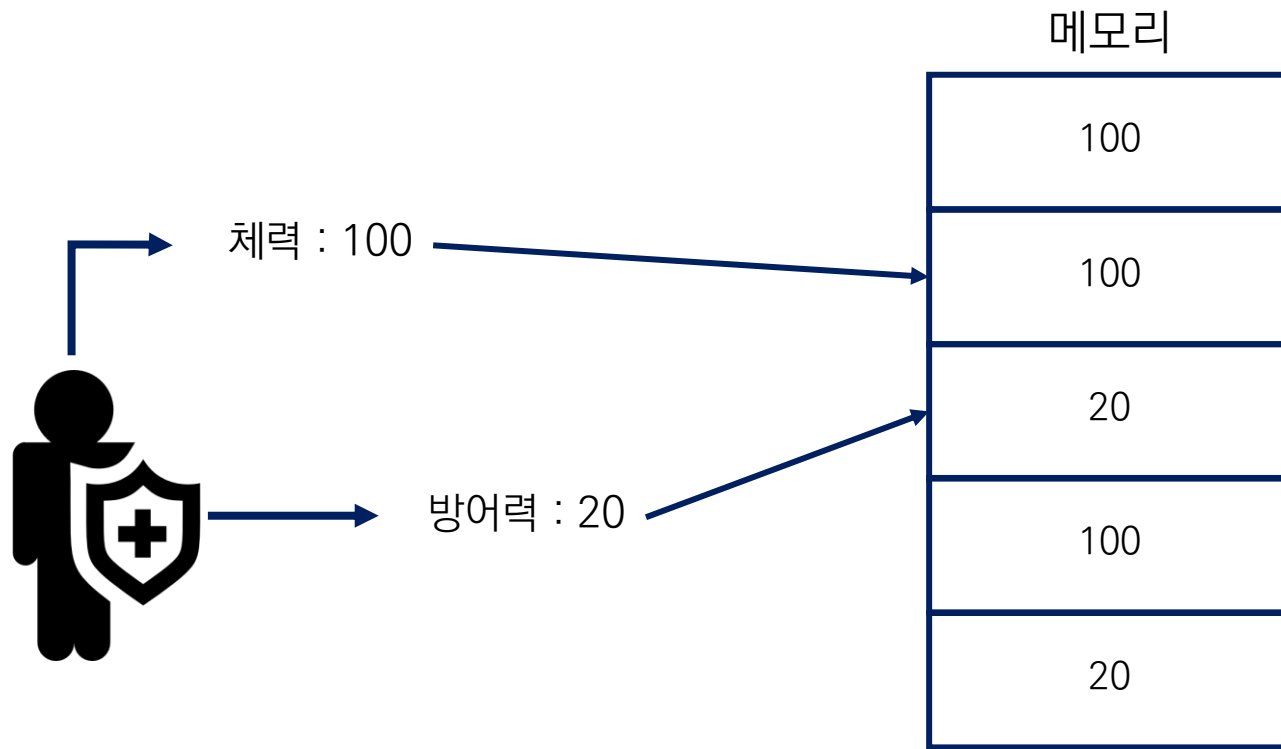
게임에 사용되는 데이터는 대부분 메모리에 저장 되어있다

게임 메모리 해킹



이 부분을 조작하면 돈 체력 등 게임에서의 주요 재화를 마음대로 할 수 있다

게임 메모리 해킹

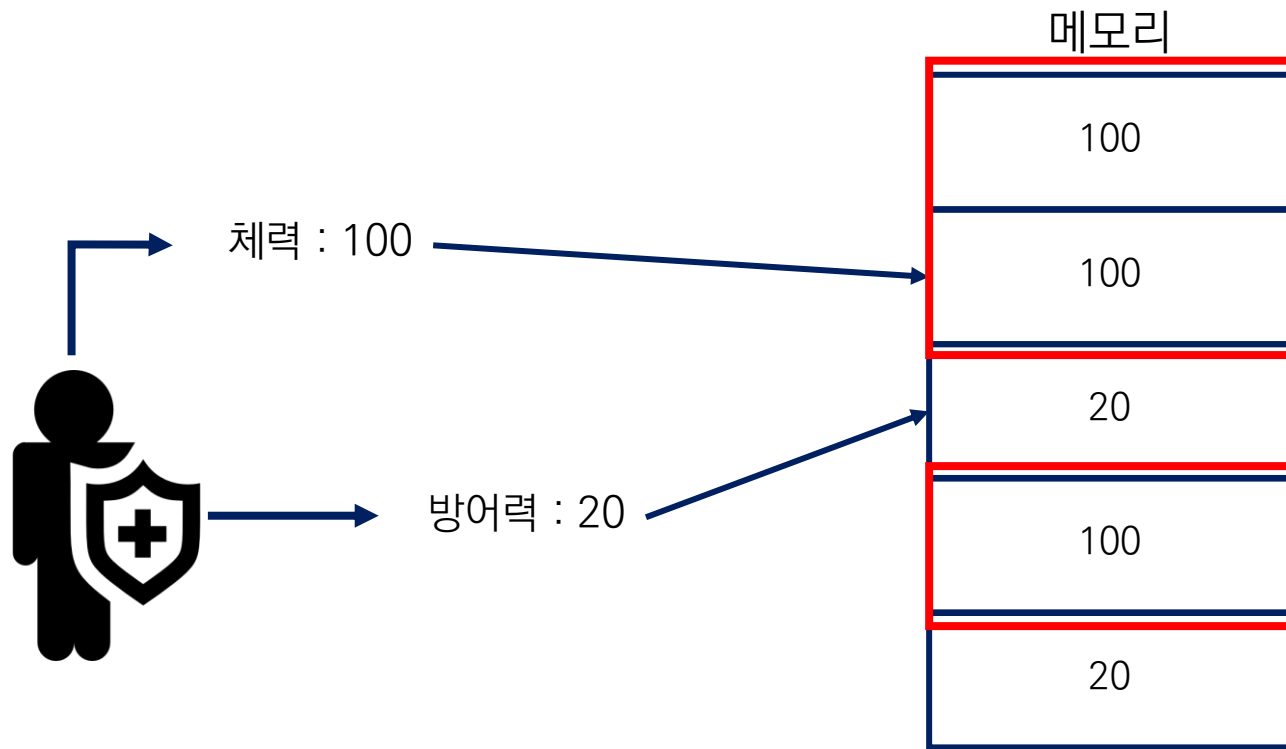


아니 뭐가 체력이누;



하지만 메모리상에 같은 숫자는 너무나도 많다.

게임 메모리 해킹

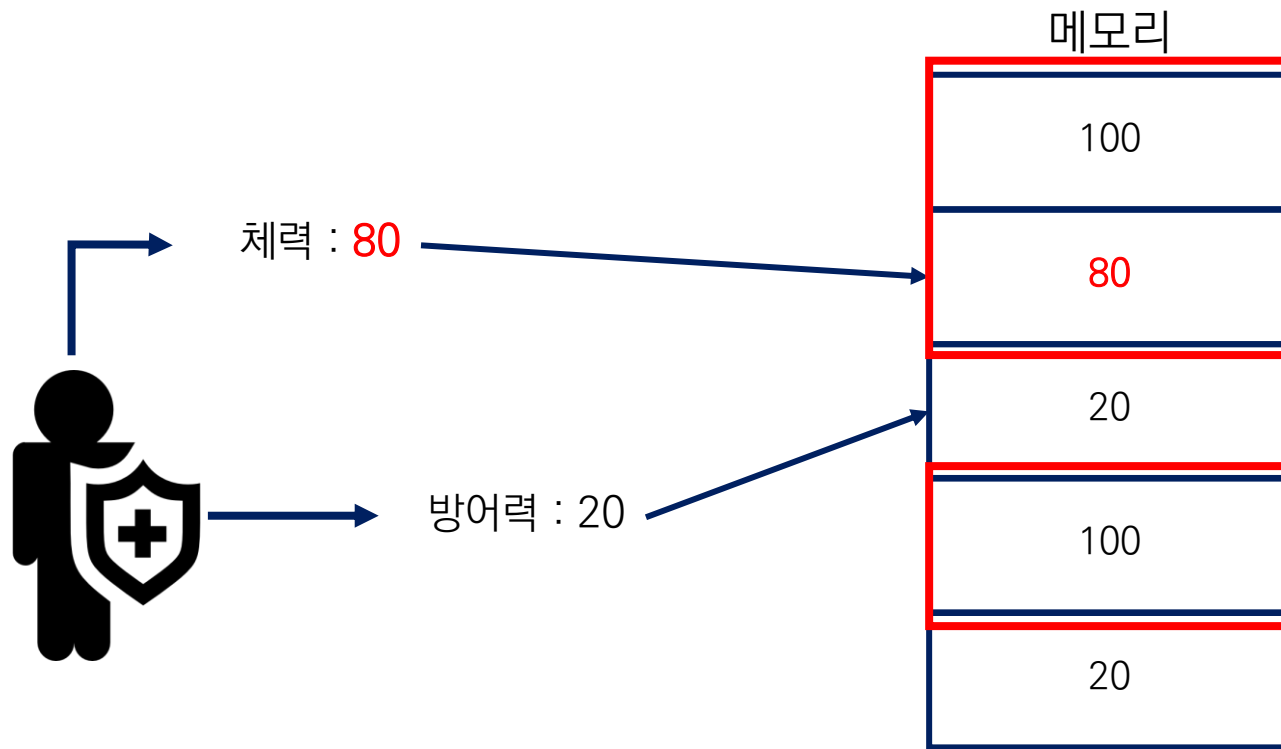


일단 싸그리 다 검색 ㄱㄱ



그래서 다음과 같은 방법으로 진행한다.
체력 검색 -> 체력 감소 -> 검색된 체력중에 감소된 체력 검색 -> ... 반복

게임 메모리 해킹

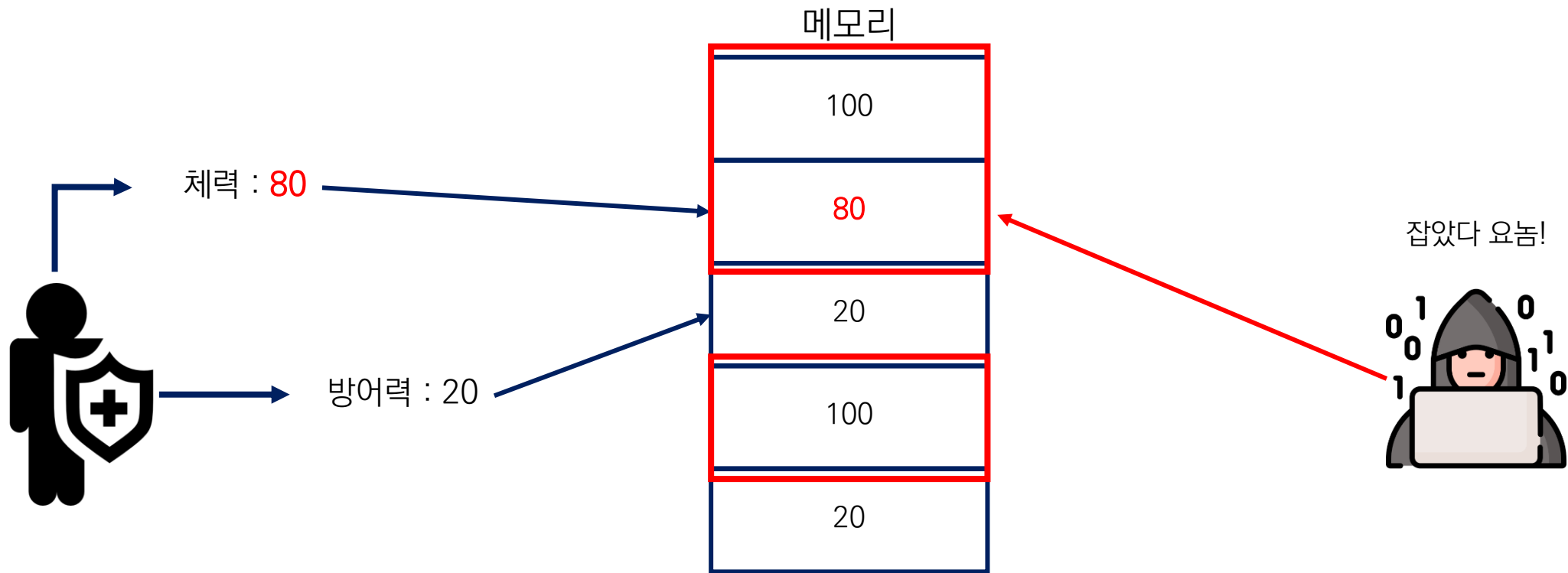


체력 감소시키기 ㄱㄱ



그래서 다음과 같은 방법으로 진행한다.
체력 검색 -> 체력 감소 -> 검색된 체력중에 감소된 체력 검색 -> ... 반복

게임 메모리 해킹



감소된 체력으로 다시 검색

게임 메모리 해킹

```
function Memory_scan()
{
    var ranges = Process.enumerateRangesSync({protection: 'r--', coalesce: true});
    var range;

    function Next_Range(){
        range = ranges.pop();
        if(!range)
        {
            console.log("Memory Scan Done!");
            return;
        }

        Memory.scan(range.base, range.size, "70 79 30 7a 7a 31", //py0zz1
        {
            onMatch: function(address, size)
            {
                console.log("[*] Pattern Found at: " + address.toString());
                console.log(hexdump(address,
                {
                    offset:0,
                    length:32
                }));

                console.log("");
            },
            onError: function(reason)
            {
                console.log("[!] Error Scanning Memory - " + reason );
            },
            onComplete: function()
            {
                Next_Range();
            }
        });
    }
    Next_Range();
}
```

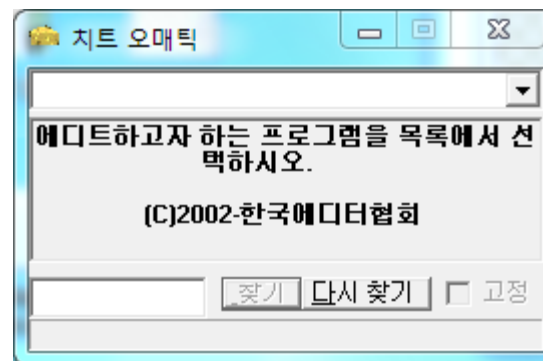
frida로 메모리 스캔하는코드? 왼쪽과 같습니다.

그런데 이거랑 똑같은 역할을 해주는 프로그램이 있어요
언제까지 노가다로 코드짤까 ^^ + 저도 귀..찮아요 ㅎ

Memory Cheating

게임 메모리 해킹

<https://cheat-engine.kr.uptodown.com/windows/download>

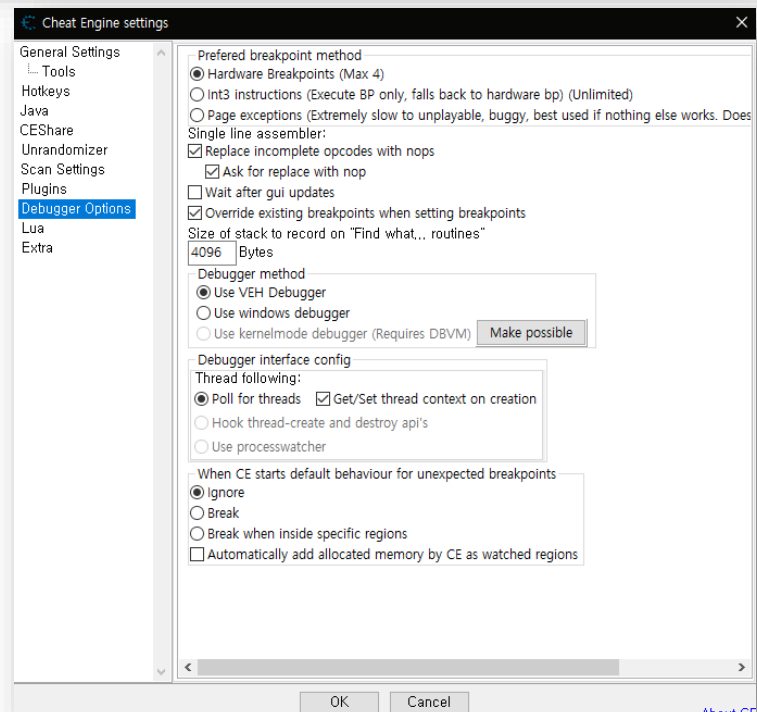
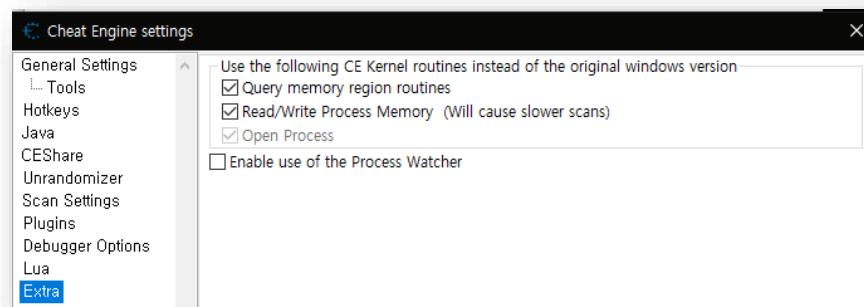
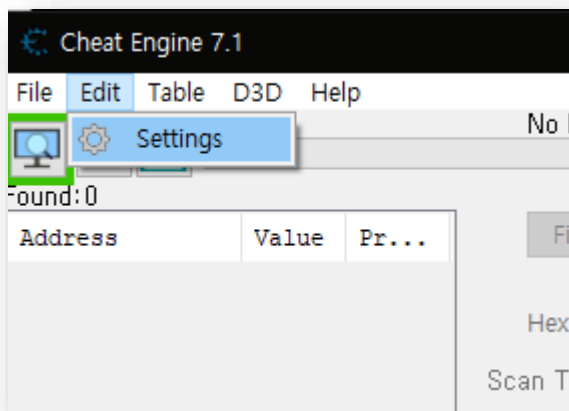


feat. 네이버 동물농장

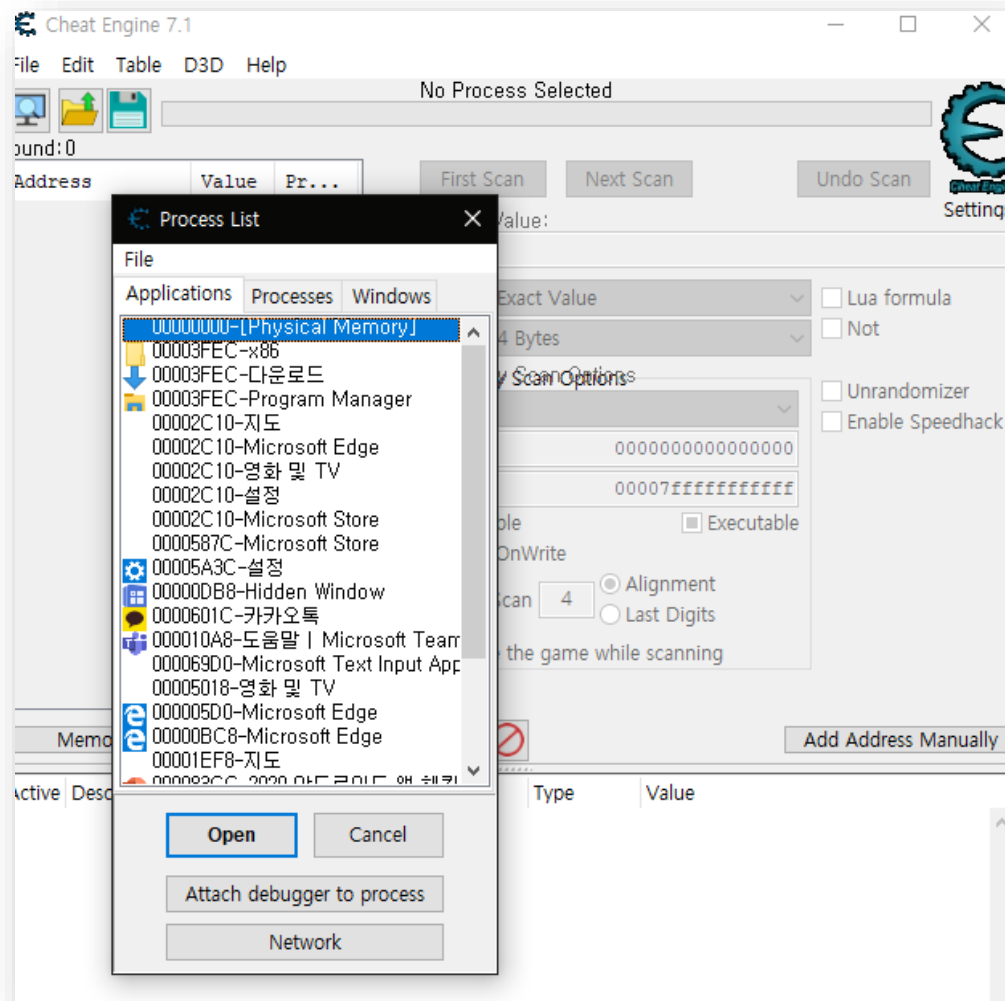
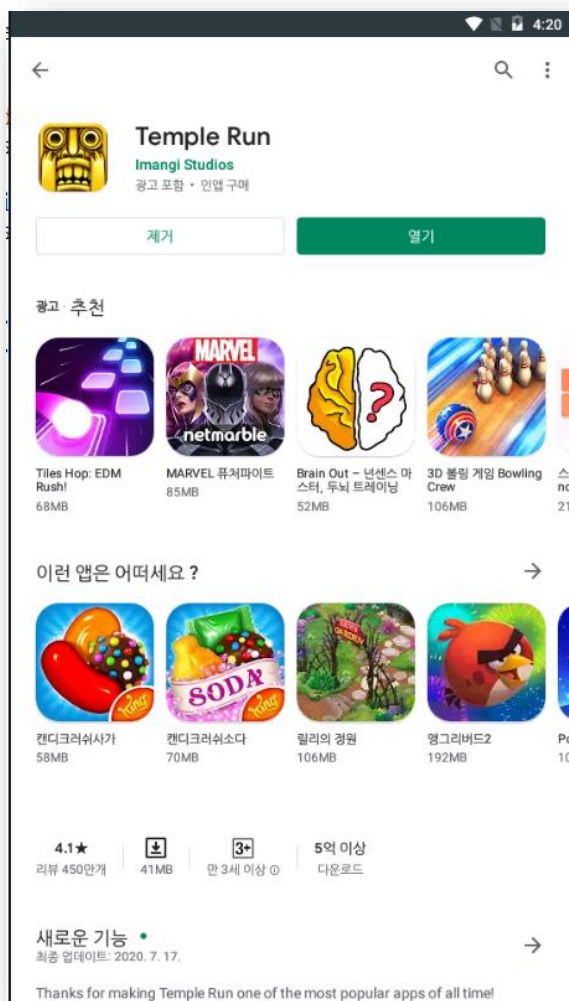
1세대 해킹툴

치트엔진 치트오매틱 해킹수준 실화냐? 정말 해킹툴 세계관 최강자들의 싸움이다.. 메모리가 웅장해진다..

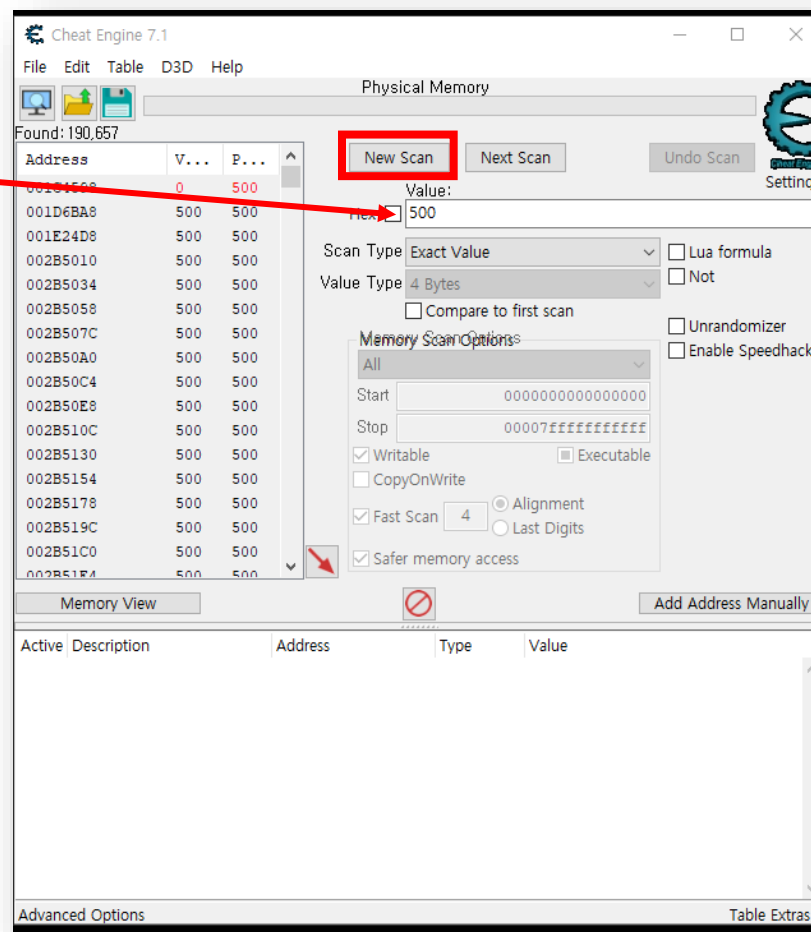
게임 메모리 해킹



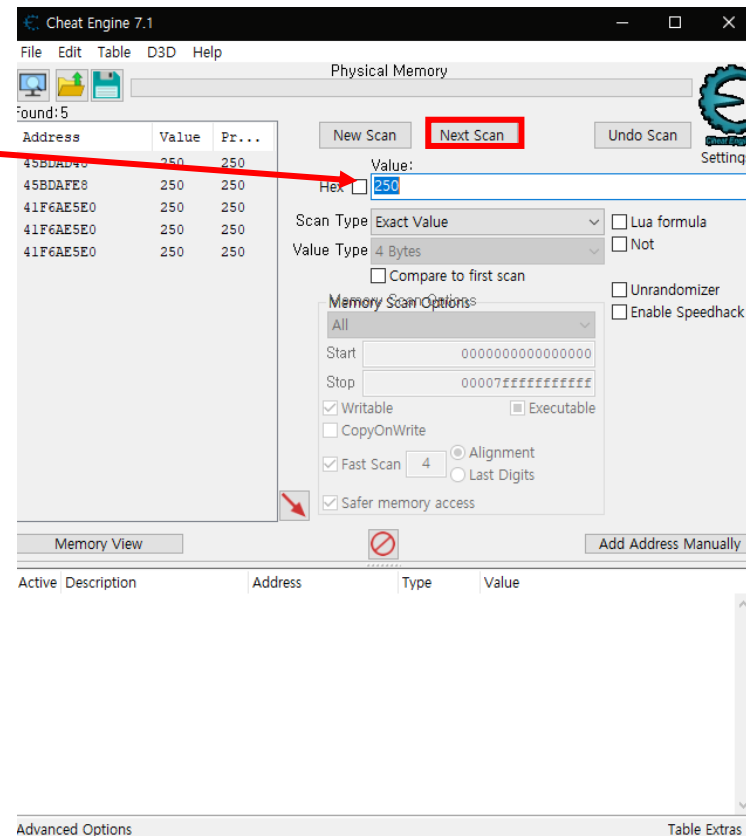
실습2 - 템플런 부자되기



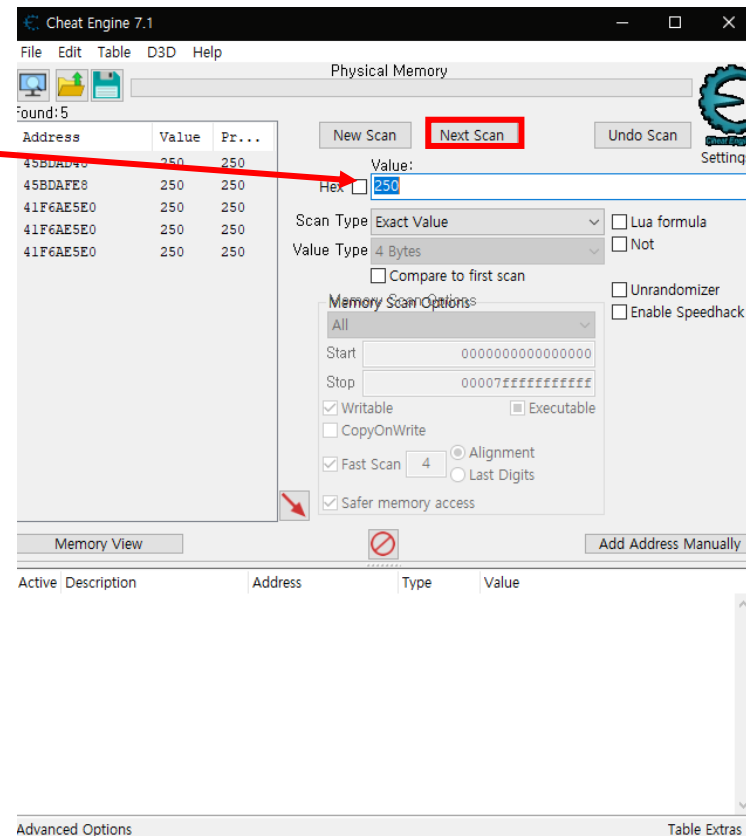
실습2 - 템플런 부자되기



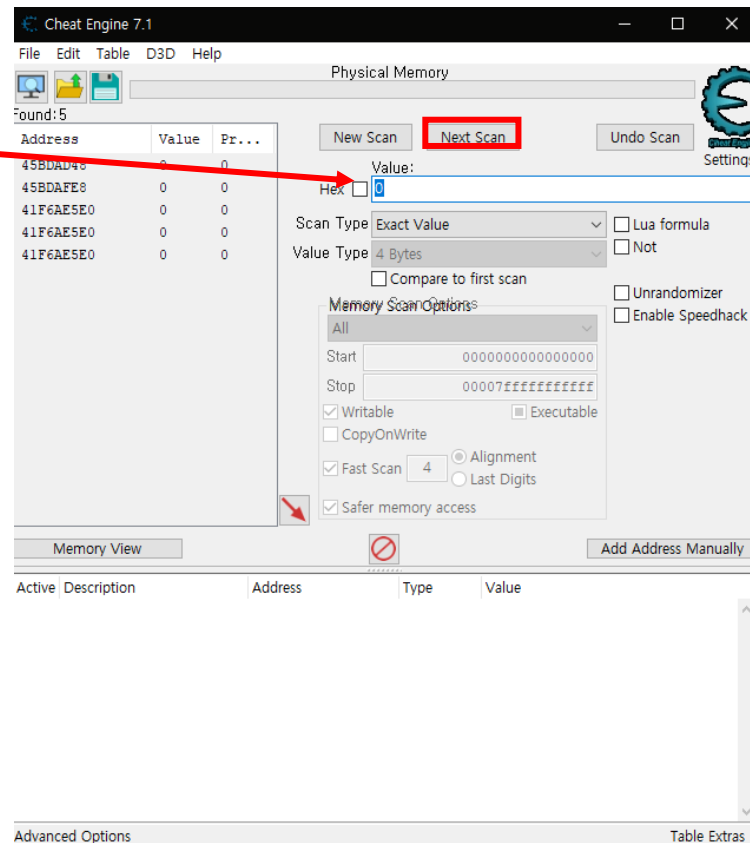
실습2 - 템플런 부자되기



실습2 - 템플런 부자되기



실습2 - 템플런 부자되기



실습2 - 템플런 부자되기

Active	Description	Address	Type	Value
<input checked="" type="checkbox"/>	No description	45BDAD48	4 Bytes	999999
<input checked="" type="checkbox"/>	No description	45BDAFE8	4 Bytes	0
<input checked="" type="checkbox"/>	No description	41F6AE5E0	4 Bytes	0
<input checked="" type="checkbox"/>	No description	41F6AE5E0	4 Bytes	0
<input checked="" type="checkbox"/>	No description	41F6AE5E0	4 Bytes	0

Change Value

what value to change this to?

OK Cancel


Active	Description	Address	Type	Value
<input checked="" type="checkbox"/>	No description	45BDAD48	4 Bytes	999999
<input checked="" type="checkbox"/>	No description	45BDAFE8	4 Bytes	999999
<input checked="" type="checkbox"/>	No description	41F6AE5E0	4 Bytes	999999
<input checked="" type="checkbox"/>	No description	41F6AE5E0	4 Bytes	999999
<input checked="" type="checkbox"/>	No description	41F6AE5E0	4 Bytes	999999

실습2 - 템플런 부자되기



실제 게임 해킹해보기(유튜브 제공 XX)

과제 - 헥사곤 던전





Hexagon Dungeon : 헥사 블록 퍼즐

Bleor Games

광고 포함 · 인앱 구매




4.4 ★
리뷰 1천개

 40MB

 만 3세 이상 ①

1만 이상
다운로드

[설치](#)



게임 소개

전략적인 퍼즐! 연속 콤보로 적들을 물리치는 쾌감! 귀여운 픽셀 그래픽의 다양한 캐릭터를 수집하는 재미까지!

[퍼즐](#)[오프라인](#)[싱글플레이어](#)[픽셀화](#)

평점 및 리뷰 ①

4.4

★★★★☆

1,080

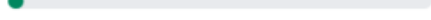
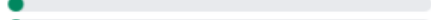
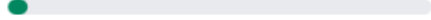
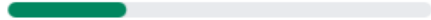

5

4

3

2

1



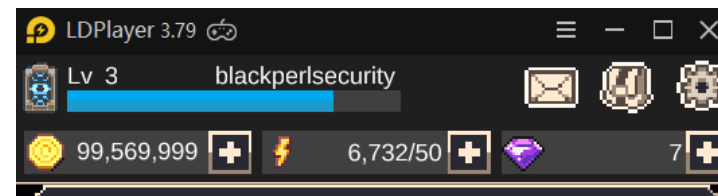
게임플레이 4.6 ★

컨트롤 4.5 ★

그래픽 4.5 ★

실제 게임 해킹해보기(유튜브 제공 XX)

과제 - 헥사곤 던전



실제 게임 해킹해보기(유튜브 제공 XX)

과제 – 헥사곤 던전

랭킹전, 보스전! 하지마세요 고소당합니다.