

# Hooking on Android



2014.07.05

정광운

[exsociety@gmail.com](mailto:exsociety@gmail.com)

[www.CodeEngn.com](http://www.CodeEngn.com)

2014 CodeEngn Conference 10



Code  Engn

# Who am I

- 정광운 EXSO (Not EXO)
- 27 years old (Single)
- CNU & Hackershool & Secu87
- Contact Me
  - <http://facebook.com/exsociety>
  - [exsociety@gmail.com](mailto:exsociety@gmail.com)
  - <http://bananapayload.org>



# What is Hooking?

## 후킹

위키백과, 우리 모두의 백과사전.

후킹(영어: hooking)은 소프트웨어 공학 용어로, 운영 체제나 응용 소프트웨어 등의 각종 컴퓨터 프로그램에서 소프트웨어 구성 요소 간에 발생하는 함수 호출, 메시지, 이벤트 등을 중간에서 바꾸거나 가로채는 명령, 방법, 기술이나 행위를 말한다. 이때 이러한 간섭된 함수 호출, 이벤트 또는 메시지를 처리하는 코드를 후크(영어: hook)라고 한다.



### Windows 커널단의 후킹

구사무열 / dual5651 / dualpage.muz.ro

Hook the Planet을 주제로 Windows 커널단의 후킹에 대한 전반적인 설명을 한다.

### Art of Hooking

박영호 / amesianx / powerhacker.net

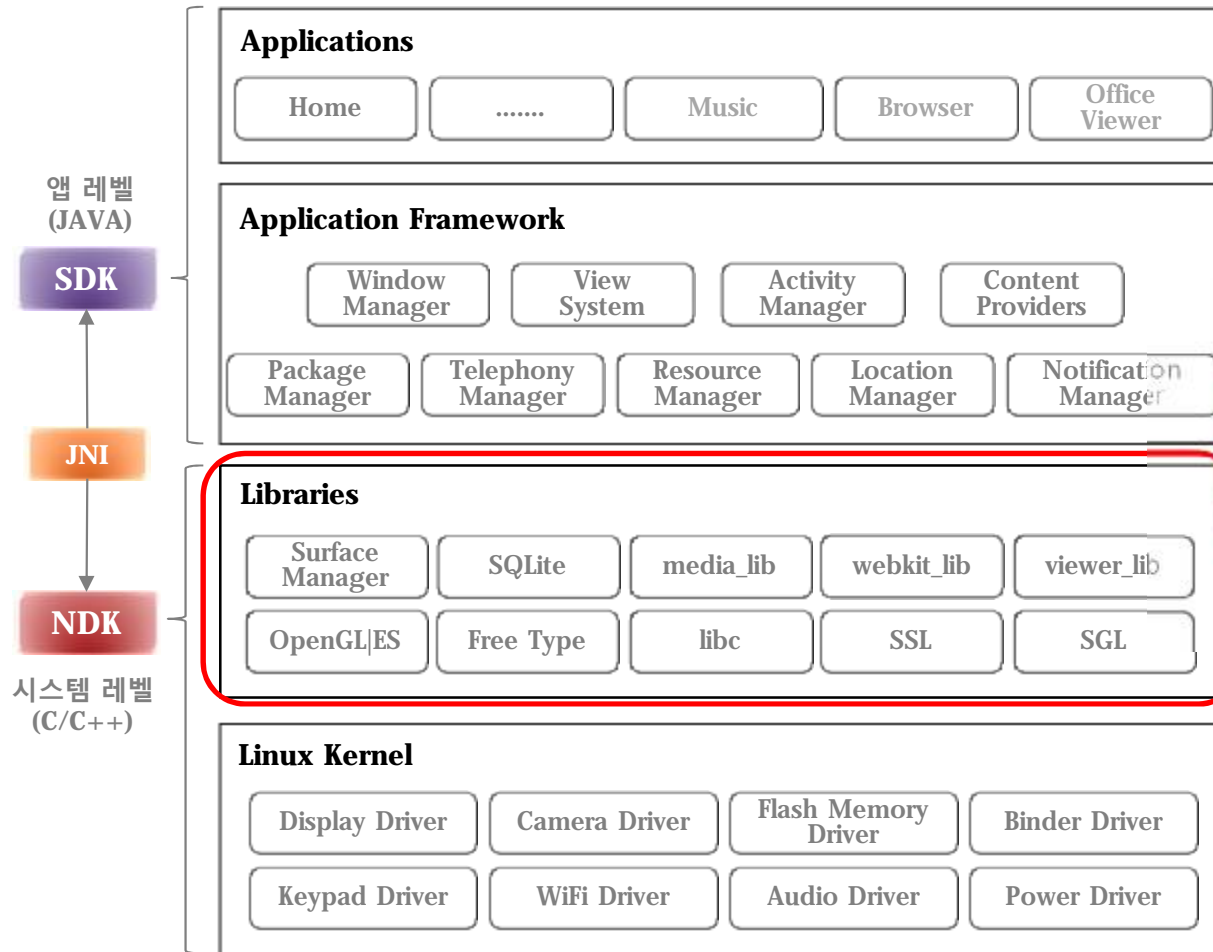
ActiveX Binary 조작 및 후킹 ActiveX Binary 조작을 하지않는 범용적 COM 후킹 키보드 후킹은 크래커가 가장 직관적으로 접근하는 해킹수단 중에 대해서 설명하고 ActiveX의 COM에 대해 알아본다. (키보드 후킹의 환경은 사용자 입력을 예상하기 힘들고 정확히 어떤 행동 중인지 포착하는 인공지능적 해킹이 어려운 점이 있다.)

### hooking and visualization

김재용 / BlueH4G

리버서플이나 어플리케이션 분석가들에게 hooking이란 말레야 필수가 없는 존재이다. 이러한 후킹을 위해 detours 등 매우 많은 라이브러리도 나와 있지만, 많은 수의 어플리케이션을 분석하거나, 심플하게 내부 클로우만 살펴보기에는 생각보다 손이 많이가는게 사실이다. 이를 좀 더 손쉽고 심플하도록 구현해 보고, visualization 을 도입하여 좀더 직관적으로 분석할 수 있도록 해 볼 것이다.

# Android System Overview



Hooking on Android

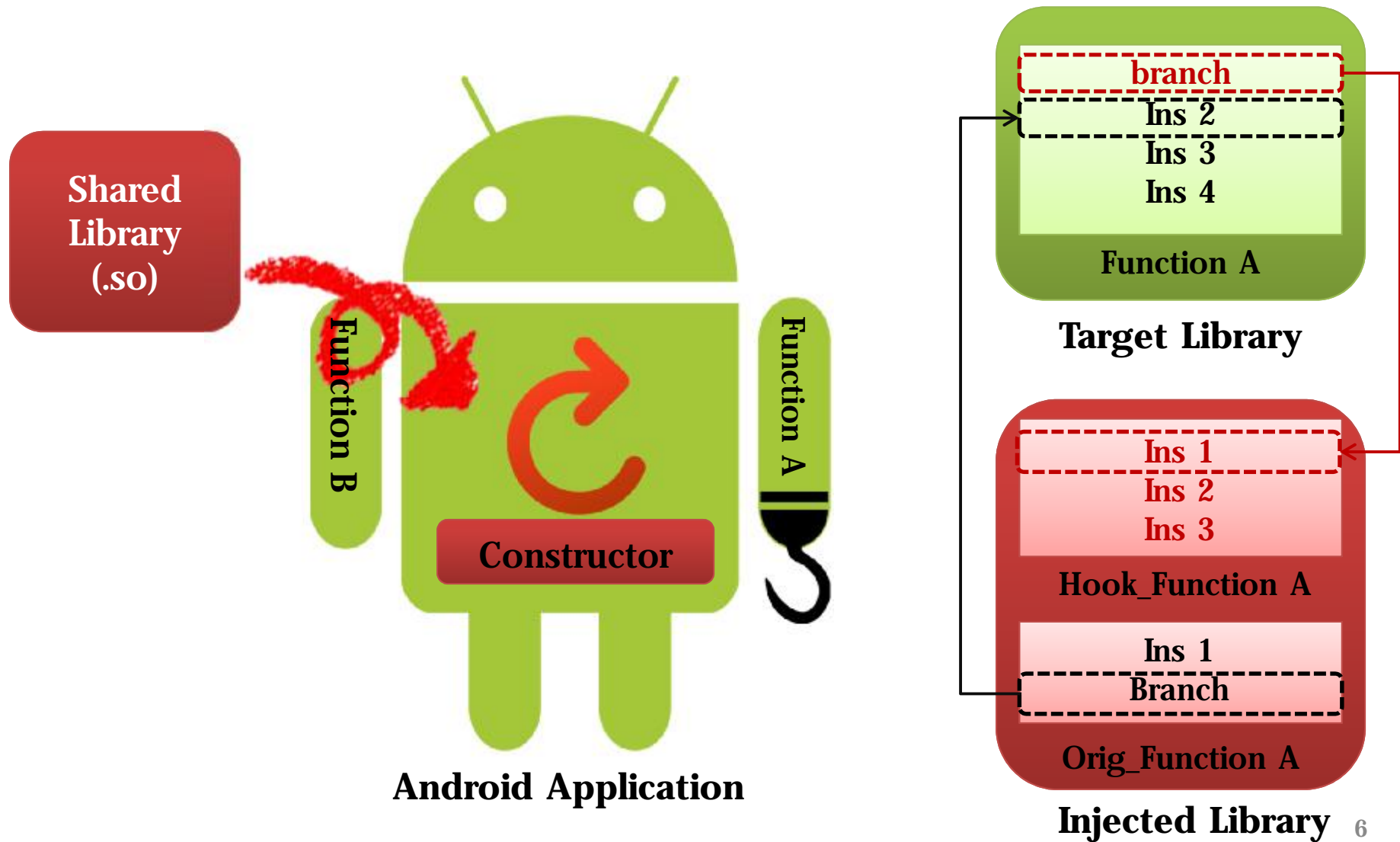
Hooking on ARM

# Goal

- ARM 기반의 안드로이드 환경
- 시스템의 수정 X (단, 루팅 필요)
- 애플리케이션의 수정 X
- 애플리케이션의 라이브러리 내 함수에 대한 후킹 수행



# Design of Hooker



# Shared Library Injection

- Call `dlopen()` using `ptrace()` on

## SYNOPSIS

```
#include <dlfcn.h>
```

```
void *dlopen(const char *filename, int flag);
```

## dlopen()

The function `dlopen()` loads the dynamic library file named by the null-terminated string `filename` and returns an opaque "handle" for the dynamic library. If `filename` is NULL, then the returned handle is for the main program. If `filename` contains a slash ("/"), then it is interpreted as a (relative or absolute) pathname. Otherwise, the dynamic linker searches for the library as follows (see `ld.so(8)` for further details):

## SYNOPSIS

```
#include <sys/ptrace.h>
```

```
long ptrace(enum __ptrace_request request, pid_t pid,  
            void *addr, void *data);
```

## DESCRIPTION

The `ptrace()` system call provides a means by which one process (the "tracer") may observe and control the execution of another process (the "tracee"), and examine and change the tracee's memory and registers. It is primarily used to implement breakpoint debugging, and system call tracing.

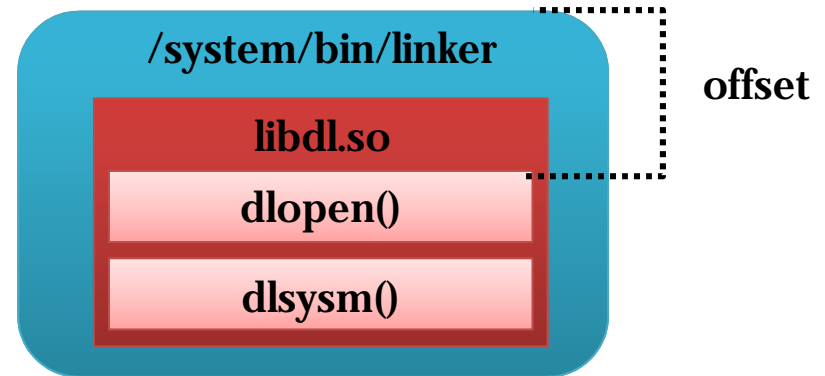
# Shared Library Injection

## 1) Find dlopen() address

Can not found libdl.so on maps

```
soinfo libdl_info = {  
    name: "libdl.so",  
    flags: FLAG_LINKED,  
  
    strtab: ANDROID_LIBDL_STRTAB,  
    symtab: libdl_syntab,  
  
    nbucket: 1,  
    nchain: 7,  
    bucket: libdl_buckets,  
    chain: libdl_chains,  
};
```

/system/bin/linker 소스코드 中



**dlopen() Address**  
= base address of linker +  
offset



## 2) write library path

- use stack

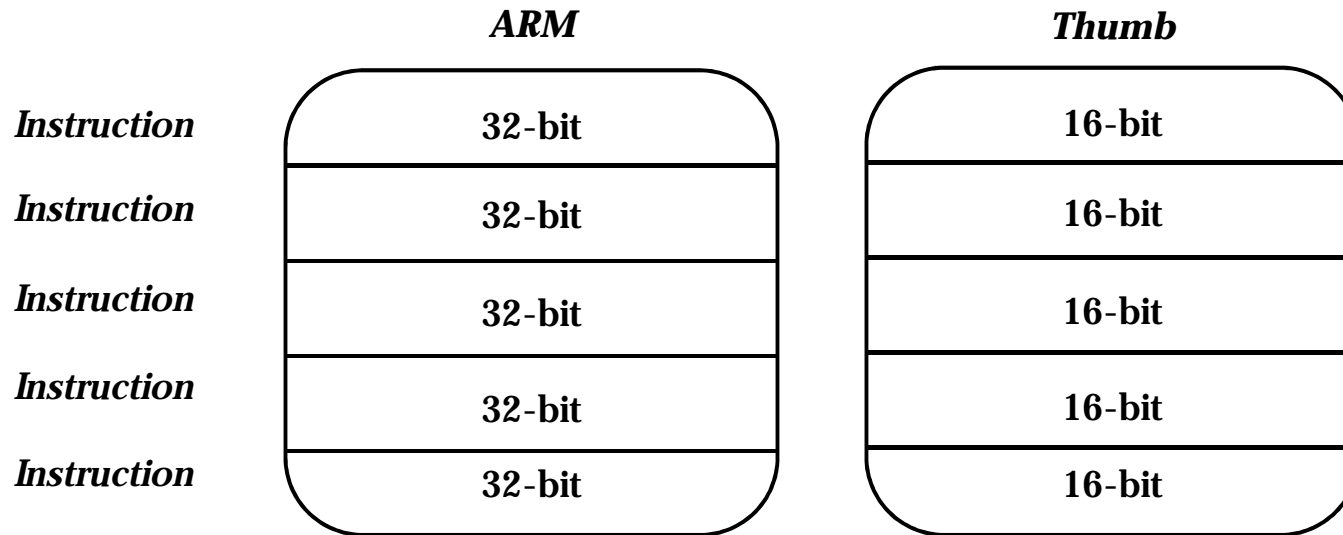
```
5a111000-5a112000 rw-p 00000000 00:00 0
5a112000-5a211000 ---p 00000000 00:00 0
be8de000-be8ff000 rw-p 00000000 00:00 0      [stack]
ffff0000-ffff1000 r-xp 00000000 00:00 0      [vectors]
```

```
0xbe8de000: 0x00000000 0x00000000 0x00000000 0x00000000
0xbe8de010: 0x00000000 0x00000000 0x00000000 0x00000000
0xbe8de020: 0x00000000 0x00000000 0x00000000 0x00000000
```

- **PTRACE\_POKEDATA**

`ptrace(PTRACE_POKEDATA, pid, dst address, 4byte_data)`

# ARM Instruction mode



**Function Address**

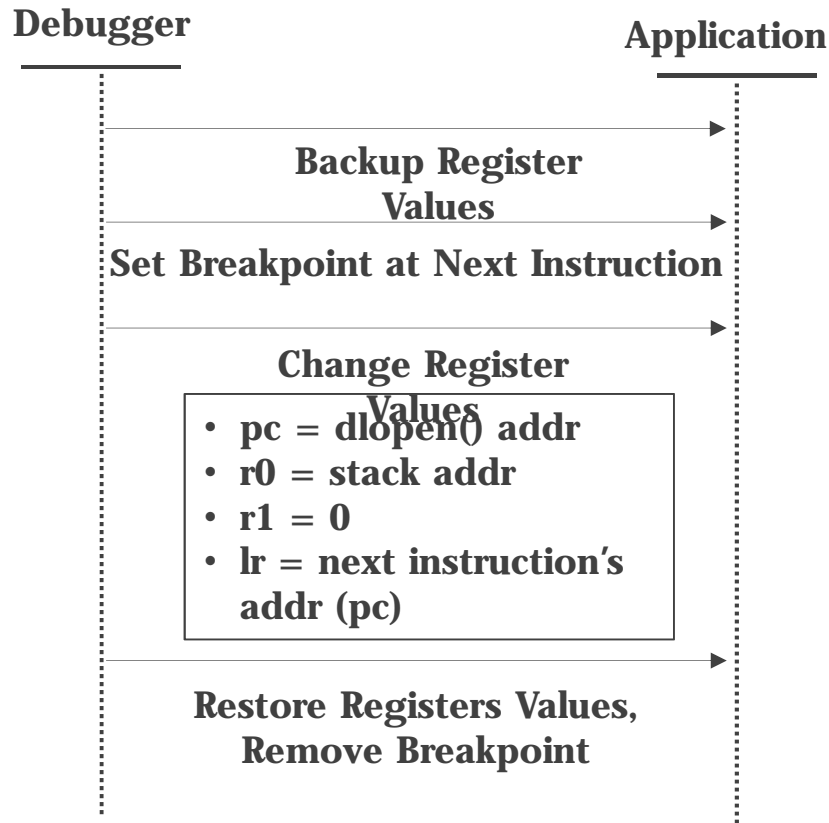


**CPSR Register**

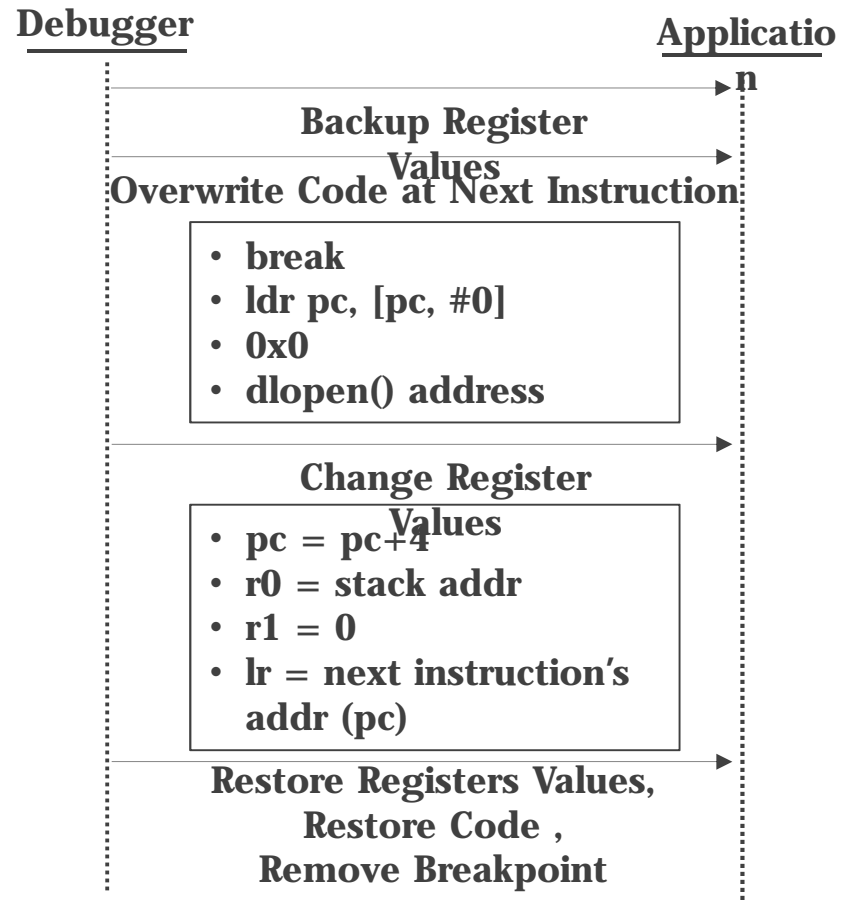
**T = 0 : ARM Mode**  
**T = 1 Thumb Mode**

### 3) Call dlopen() **β** Thumb

**Thumb 모드**



**ARM 모드**



## 4) result

Useage : injector [pid] [Library Full Path]

```
bash-4.2# ./injector 30931 /data/local/root/hook/libfoo.so
Attach Success! Pid : 30931
Call dlopen(/data/local/root/hook/libfoo
```

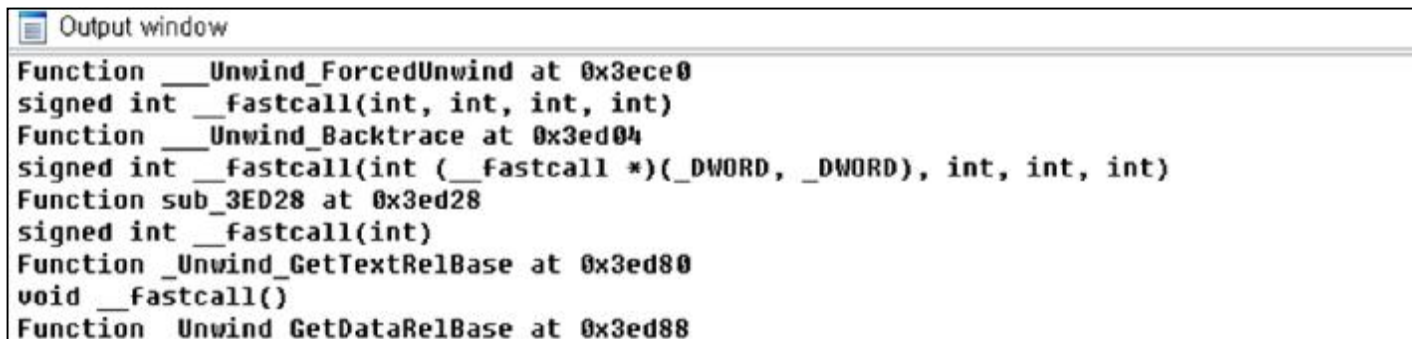
```
79382000-79393000 rw-p 00000000 00:00 0
79393000-793ad000 rw-p 00000000 00:00 0
793ad000-793af000 rw-s 00034000 00:0b 6285
793af000-793c1000 rw-p 00000000 00:00 0
793c1000-793e1000 rw-p 00000000 00:00 0
793e1000-79411000 rw-p 00000000 00:00 0
79411000-79440000 rw-p 00000000 00:00 0
79440000-79441000 r-xp 00000000 00:00 1c 629086 /data/local/root/hook/libfoo.so
79441000-79448000 ---p 00000000 00:00 0
79448000-79449000 rw-p 00000000 00:00 1c 629086 /data/local/root/hook/libfoo.so
79449000-7944e000 rw-p 00000000 00:00 0
7944e000-79455000 rw-p 00000000 00:00 0
79472000-7947b000 rw-p 00000000 00:00 0
```

끝

# Function Hooking

## 1) Find function information

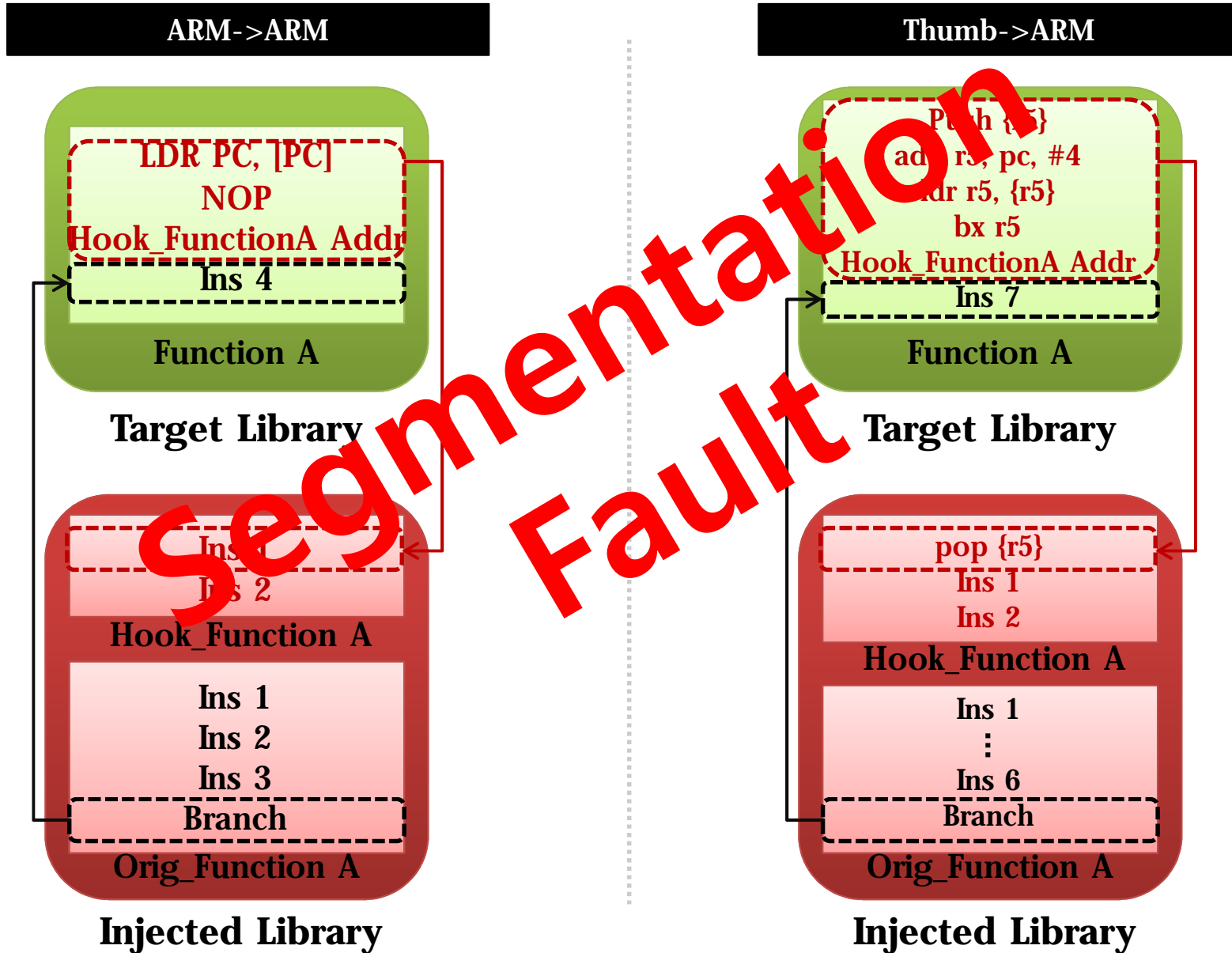
- Reference Header file
- Use Hex-ray

A screenshot of a software window titled "Output window" containing a list of function names and their addresses. The text is as follows:

```
Function __Unwind_ForcedUnwind at 0x3ece0
signed int __fastcall(int, int, int, int)
Function __Unwind_Backtrace at 0x3ed04
signed int __fastcall(int (__fastcall *)(_DWORD, _DWORD), int, int, int)
Function sub_3ED28 at 0x3ed28
signed int __fastcall(int)
Function __Unwind_GetTextRelBase at 0x3ed80
void __fastcall()
Function __Unwind_GetDataRelBase at 0x3ed88
```

<https://github.com/EiNSTeiN-/hexrays-python>

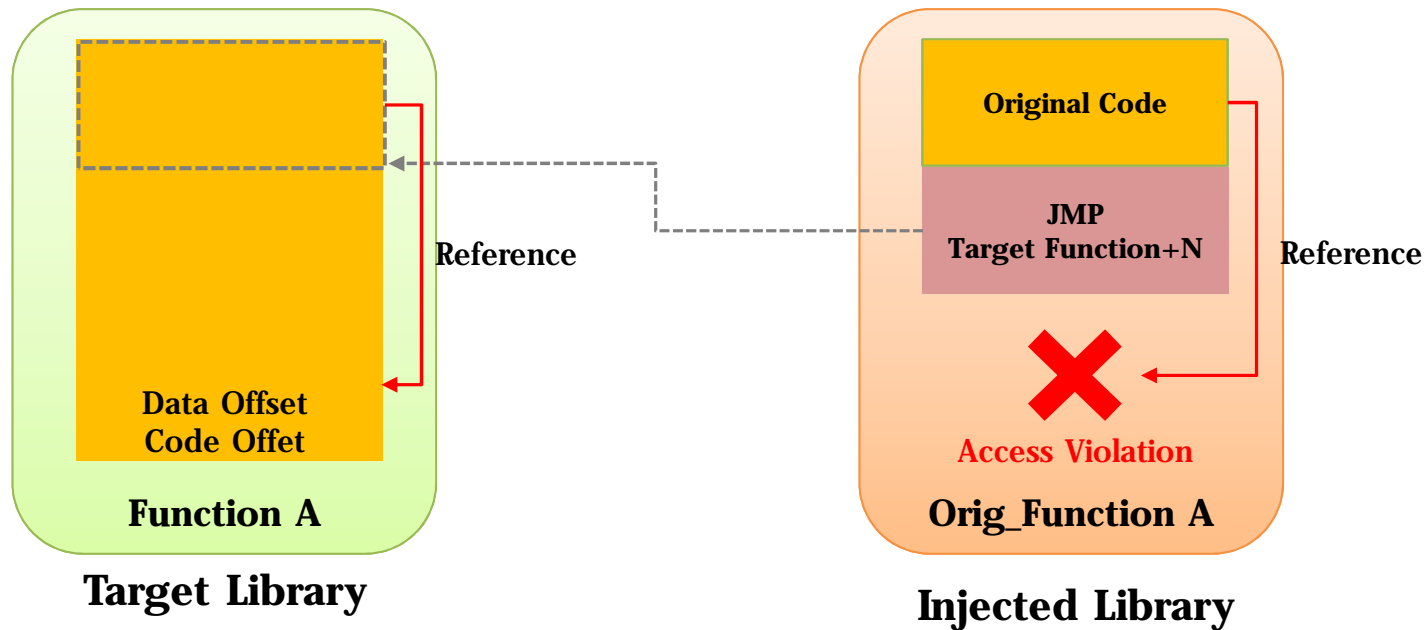
## 2) Install Hooker



# Why...?

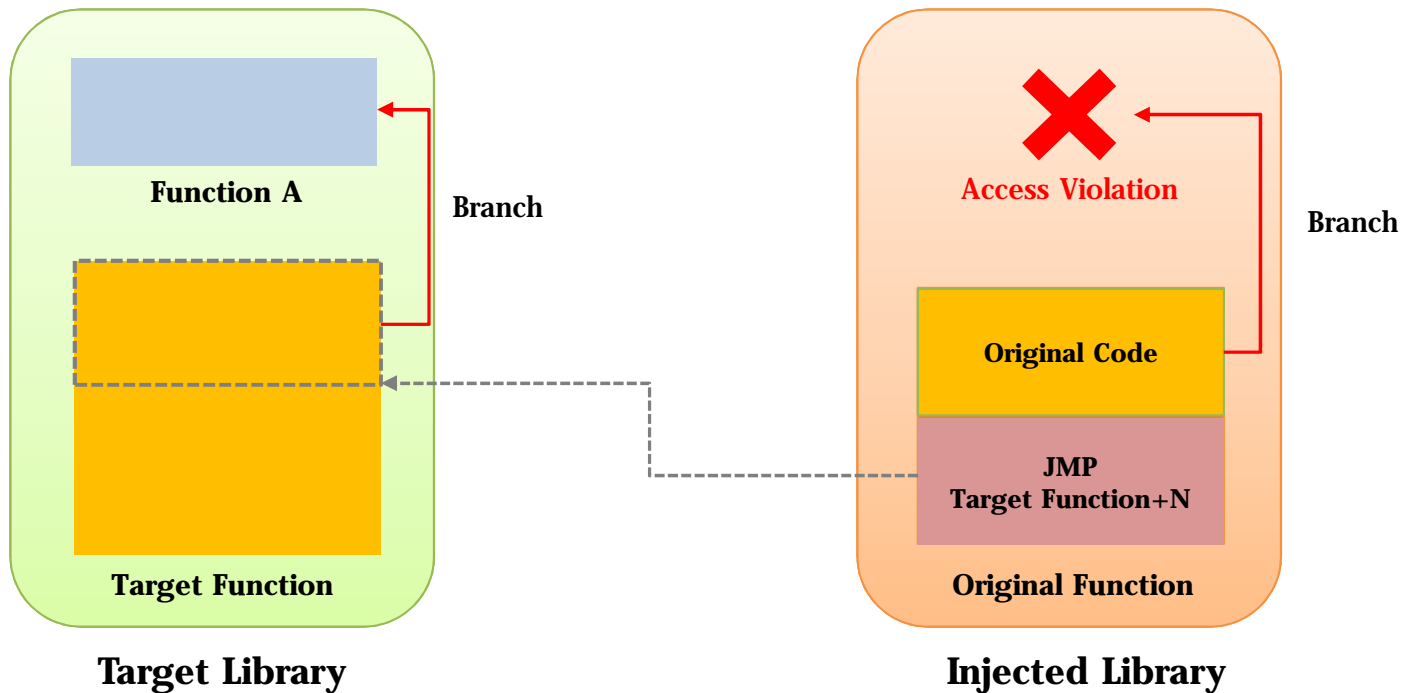
- Internal memory reference

```
d9a0 <malloc>:  
d9a0:    b508    push   {r3, lr}  
d9a2:    4b03    ldr    r3, [pc, #12]  
d9a4:    447b    add    r3, pc  
d9a6:    6819    ldr    r1, [r3, #0]  
d9a8:    680a    ldr    r2, [r1, #0]
```



- External memory reference

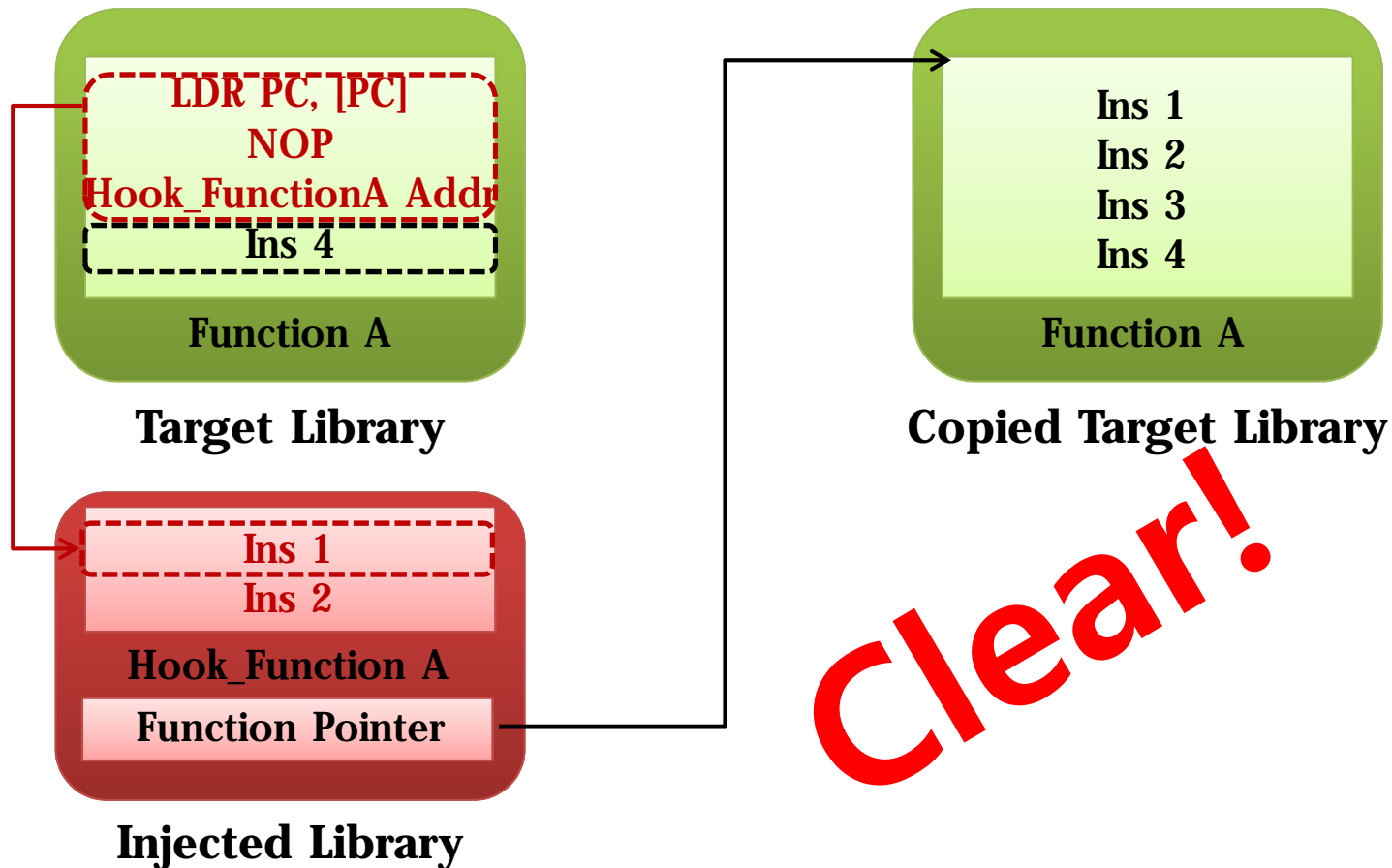
```
0000dffc <pthread_mutex_lock>:  
dffc: eaffff90 b de44 <pthread  
e000: e92d4070 push {r4, r5, r6, 1  
e004: e2504000 subs r4, r0, #0  
e008: e24dd008 sub sp, sp, #8
```





# Solution

- 귀찮으니 그냥 복사하자...



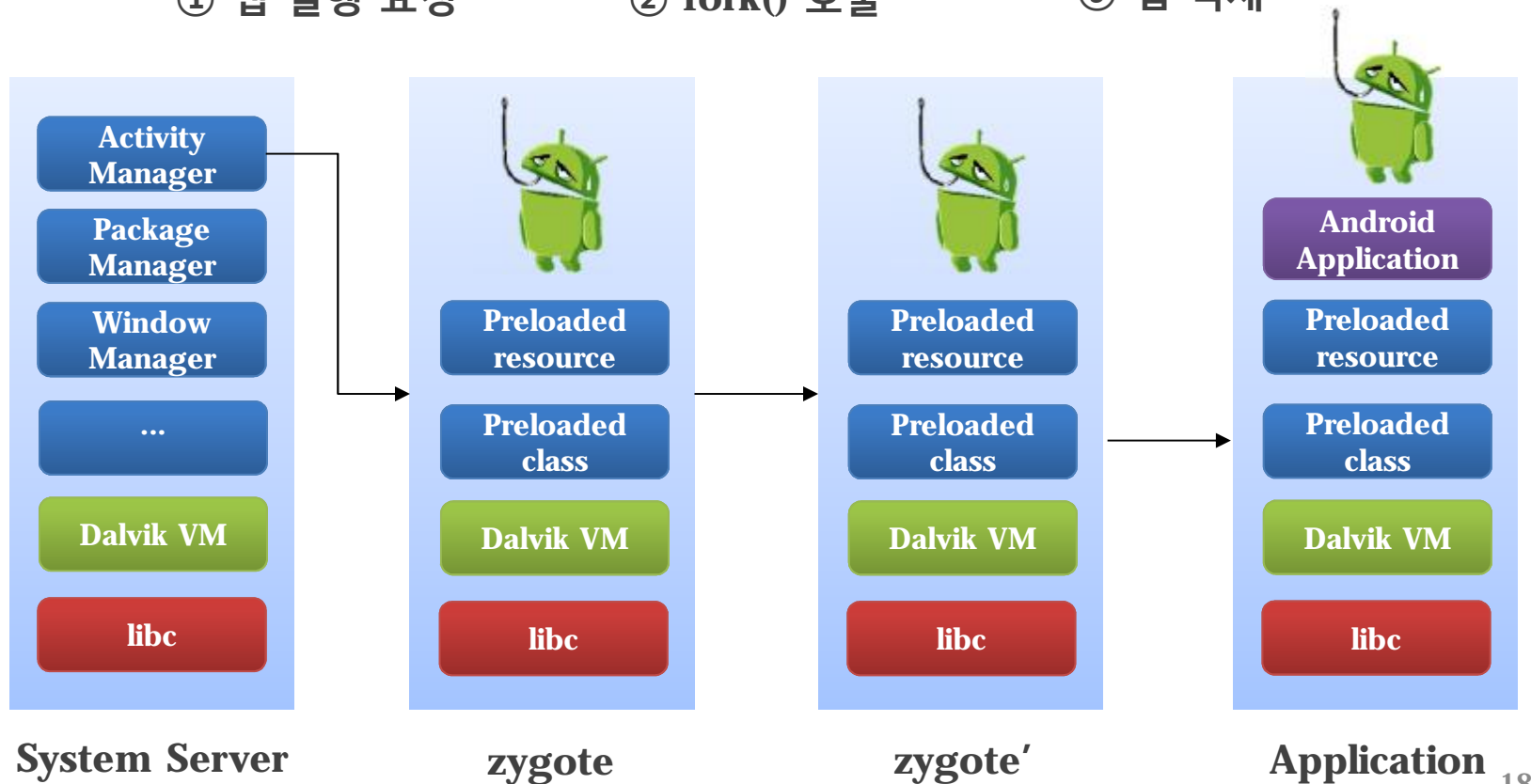
# Issue 1. Global Hook

- 애플리케이션 생성 과정

① 앱 실행 요청

② fork() 호출

③ 앱 적재



## Issue 2. 대상 라이브러리가 로드되기 전..

- 라이브러리 로드 함수를 후킹
  - `dlopen() = 10byte` // 최소 12바이트 필요
  - `dvmLoadNativeCode(char const*, Object*, char**)`

PID	TID	Tag	Text
11936	11936	HOOK	Install Success
11936	11936	HOOK	/data/app-lib/kr.co.hancom.hancomviewer.androidma □ rket-2/libhancomgraphics-4.4.3.so
11936	11936	HOOK	/data/app-lib/kr.co.hancom.hancomviewer.androidma □ rket-2/libhancomofficeengine.so

- `dvmLoadNativeCode` 종료 시점에 추가적인 Hooker 설치

# How to use

- download : <http://bananapayload.org>

```
[library path] [Name / Offset] [Function Type]  
/system/lib/libc.so malloc void *malloc(size_t size)  
/system/lib/test.so 0x400 void sub400(int, int)
```

**Define Format**



```
./ genLibrarySource [define File] [output path]
```

**Source Code**



**Edit Source & Edit makefile & make**

**library**



```
Useage : injector [pid] [Library Full Path]
```

**Hook Success**

Demo

“ 참 힘들고... 긴 시간이었지...”

QnA

뭐라고요?

[www.CodeEngn.com](http://www.CodeEngn.com)

2014 CodeEngn Conference 10

Code Engn