R.A.R.A.

Run Application as Root for Android

Jungjin Kim

2024.08.18

POC SECURITY

Root 권한으로 앱을 실행하는 방법

안드로이드 5.0 이전

AndroidManifest.xml

```
<user-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<user-permission android:name="android.permission.ACCESS_WIFI_STATE" />
…
<user-permission android:name="android.permission.ACCESS_SUPERUSER" />
```

AndroidManifest.xml 파일에 슈퍼유저 권한을 요구하는 것으로 가능

POC SECURITY

Root 권한으로 앱을 실행하는 방법

안드로이드 5.0 이후

Android 5.0부터 android.permission.ACCESS_SUPERUSER가 제거 됨

X.4. ACCESS_SUPERUSER permission DEPRECATED

Due to changes in Android 5.0 Lollipop, this permission has been deprecated and is completely ignored from SuperSU v2.30 onwards

From SuperSU version 1.20 and onwards, the *android.permission.ACCESS_SUPERUSER* permission is declared by SuperSU. All root apps should from now on declare this permission in their *AndroidManifest.xml*:

```
<uses-permission android:name="android.permission.ACCESS_SUPERUSER" />
```

If this permission is not present, SuperSU will present a warning in its superuser request popup (this is configurable in SuperSU settings). At the time of this writing this permission is not enforced, but it is expected that sometime in the future it will be, and apps requesting root that do not have this permission set will be *silently* denied.

If this permission is declared, the user will be able to see in the app permissions list that the app requests superuser access.

<http://su.chainfire.eu/#updates-permission>

POC SECURITY

How?

안드로이드 5.0 버전 이후에서 애플리케이션을
Root 권한으로 실행할 수 있을까?

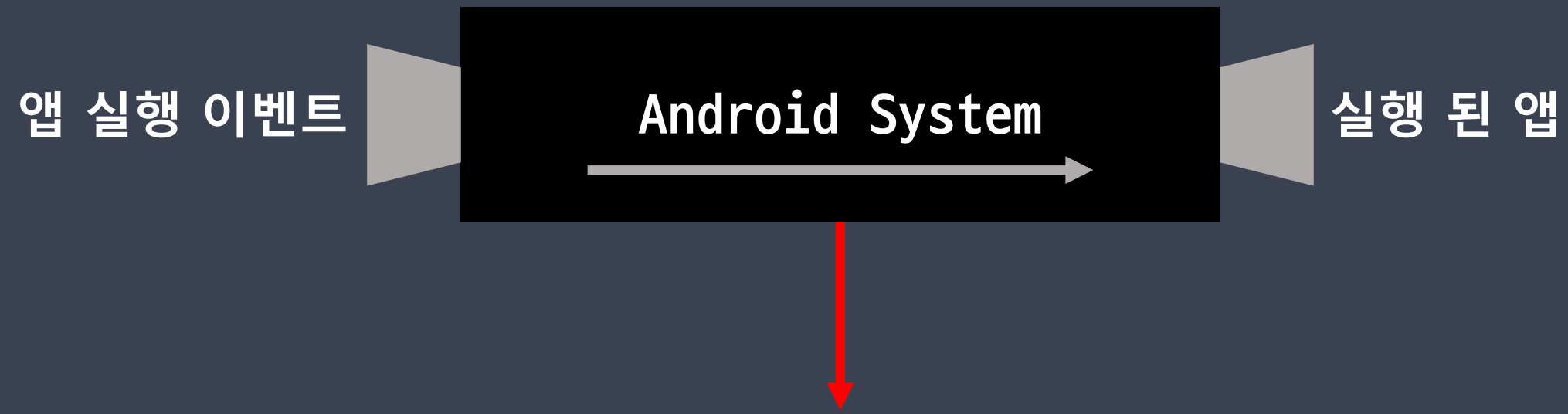
POC SECURITY

How applications run on Android

Background Knowledge

POC SECURITY

안드로이드 앱의 실행 원리



Root 권한으로 애플리케이션 실행을 위해서는
안드로이드 애플리케이션의 실행 과정 및 원리를 알아야 함

POC SECURITY

안드로이드 앱의 실행 원리

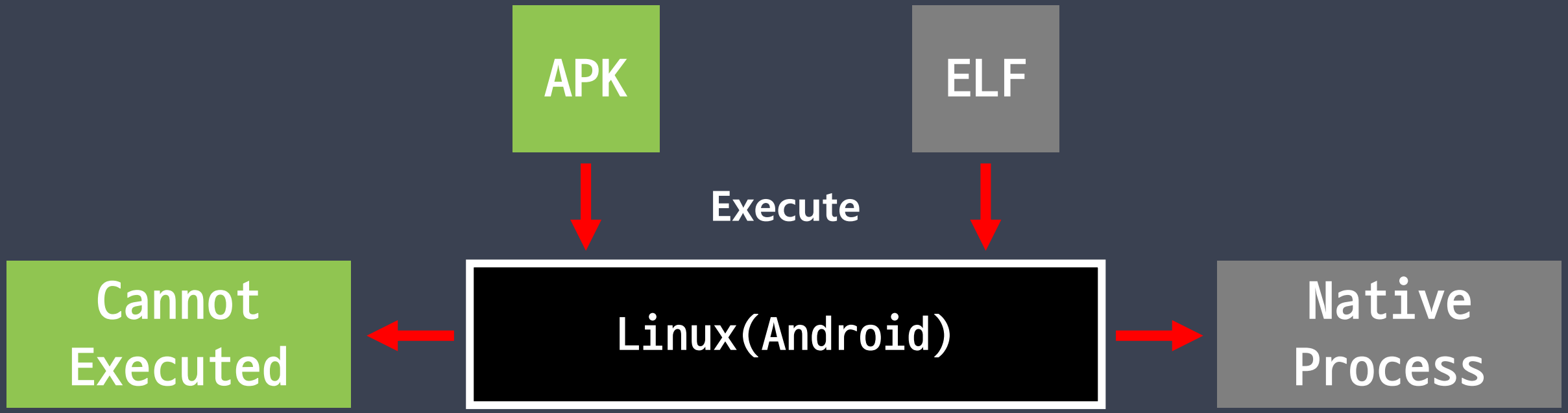
애플리케이션은 어떤 언어로 작성되어 있나?

주로 Java로 작성되어 있음

POC SECURITY

안드로이드 앱의 실행 원리

애플리케이션은 어떤 언어로 작성되어 있나?

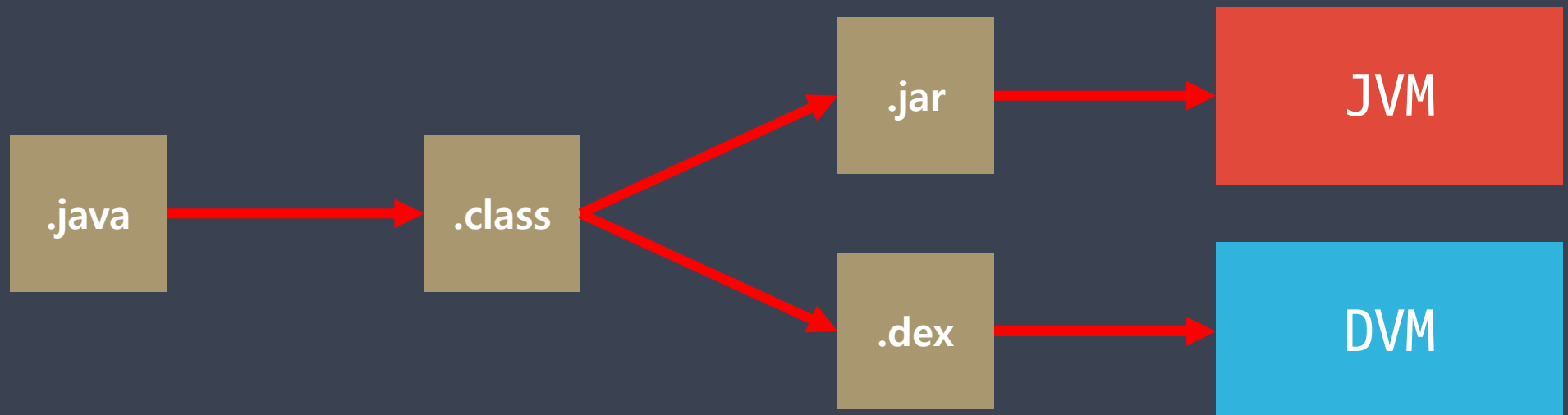


POC SECURITY

안드로이드 앱의 실행 원리

Dalvik

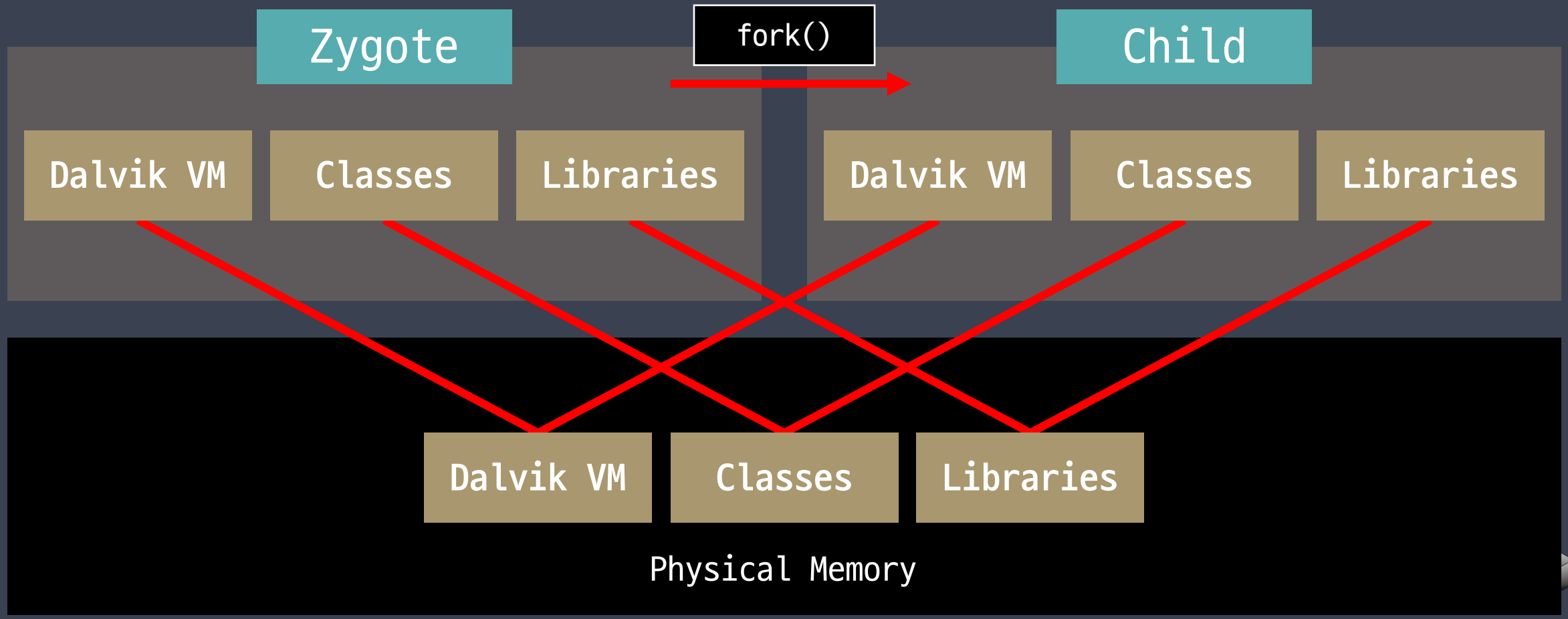
- Dalvik은 VM이며 **DVM**(Dalvik Virtual Machine)이라고 불림
- Android는 라이선스와 메모리 효율성 등의 문제로 JVM을 사용하지 않음
- JVM, DVM 모두 Java를 사용하는 VM이기에 애플리케이션을 동작하는데 문제는 없음



POC SECURITY

안드로이드 앱의 실행 원리

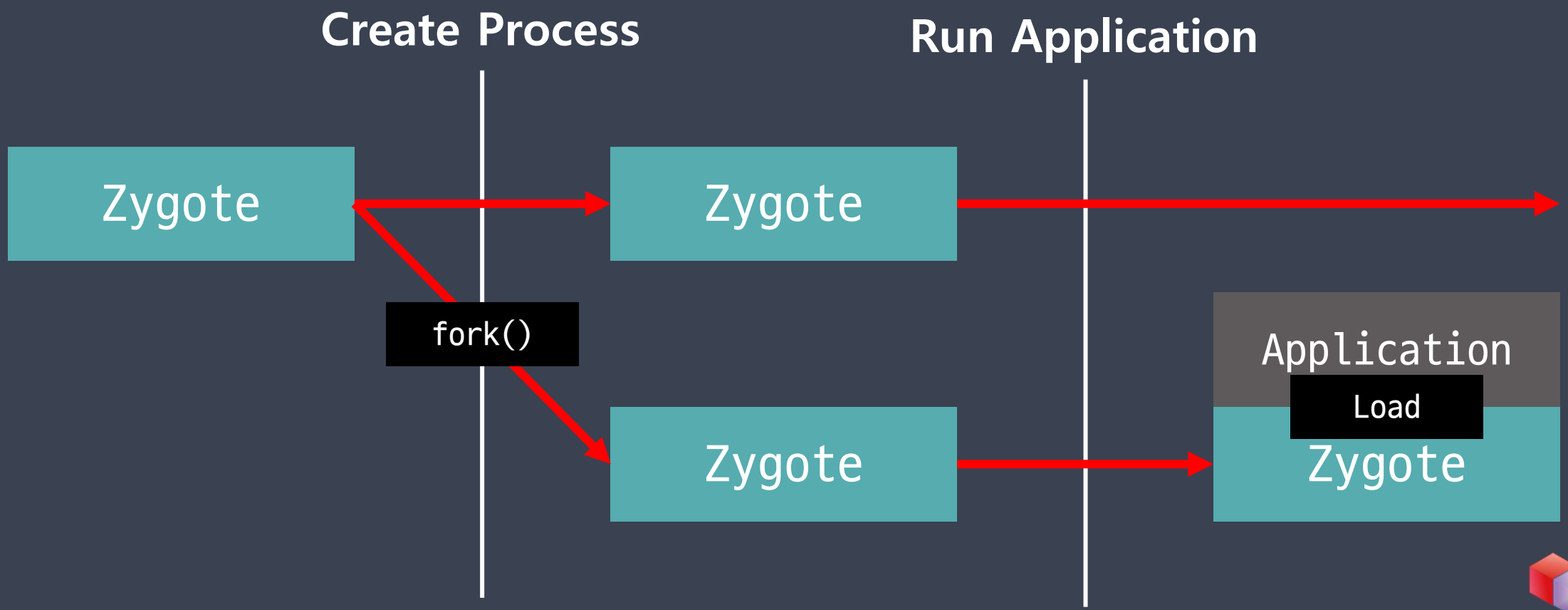
Zygote



POC SECURITY

안드로이드 앱의 실행 원리

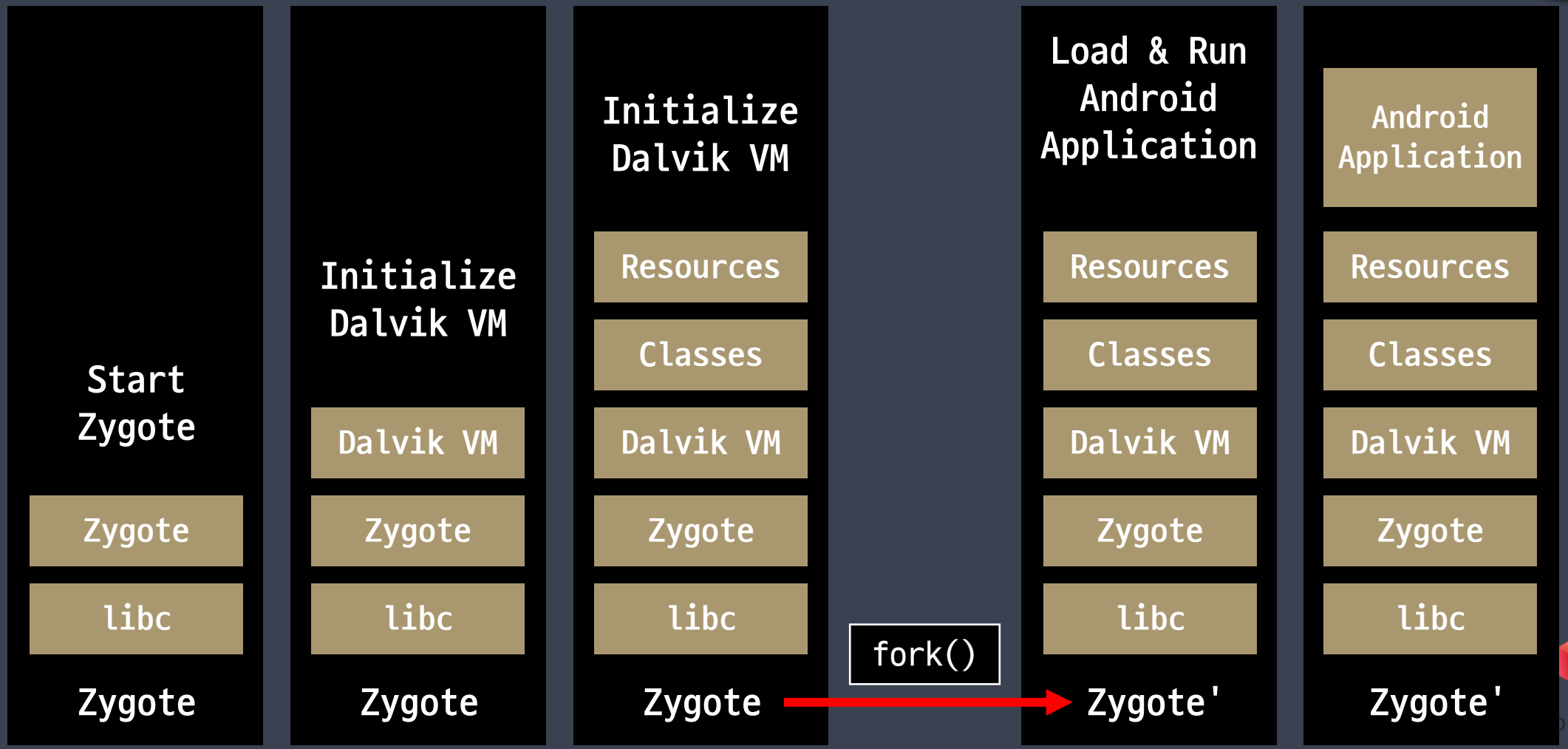
Zygote를 통한 프로세스 생성(애플리케이션 실행)



POC SECURITY

안드로이드 앱의 실행 원리

Zygote를 통한 프로세스 생성(애플리케이션 실행)



안드로이드 앱의 실행 원리

app_process란 무엇인가?

- app_process는 Zygote의 진짜(Original) 이름
- Zygote 또한 Java로 작성되어 있어 init 프로세스에서 바로 실행될 수 없음
- Zygote는 app_process를 통해 Dalvik VM을 초기화 후
  ZygoteInit 클래스를 로딩 및 실행

POC SECURITY

안드로이드 앱의 실행 원리

app_process란 무엇인가?

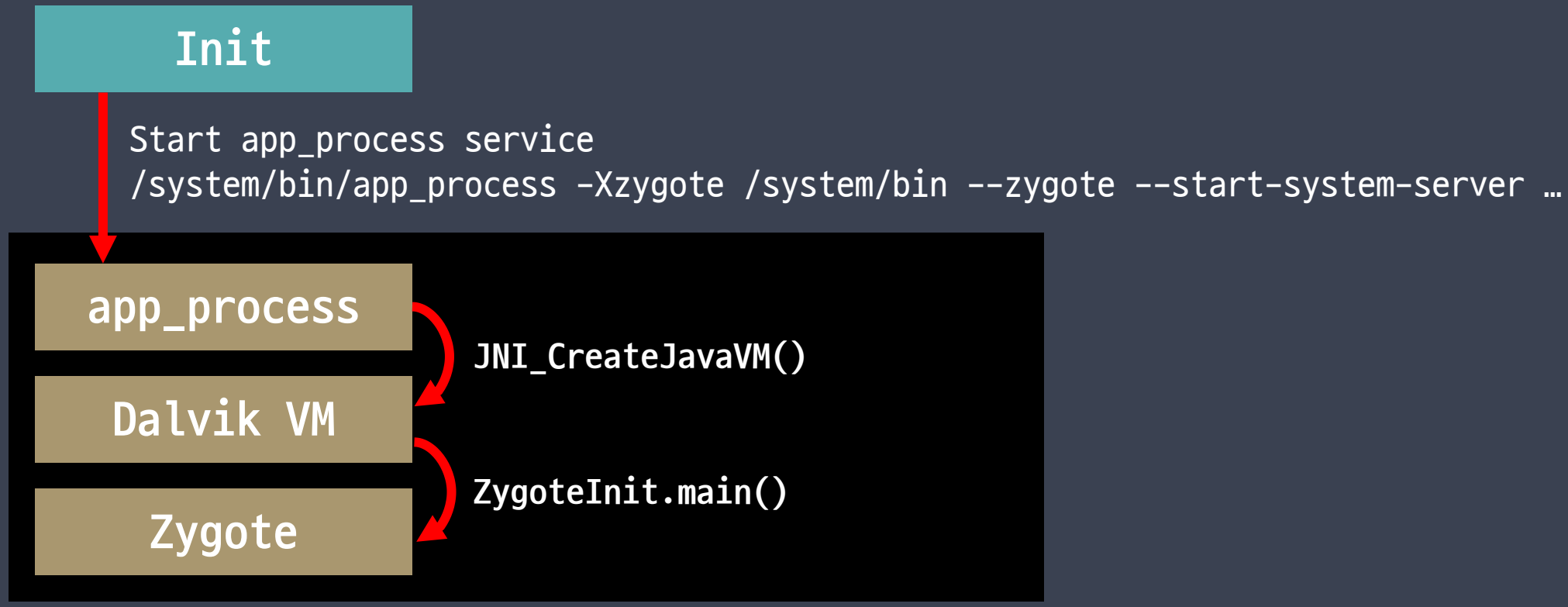
- Zygote 서비스를 실행하는 명령은 `/init.zygote64_32.rc` 파일에서 확인할 수 있음

```
service zygote /system/bin/app_process64 -Xzygote /system/bin --zygote --start-system-server --socket-
name=zygote
    class main
    socket zygote stream 660 root system
    onrestart write /sys/android_power/request_state wake
    …
    onrestart restart netd
    writepid /dev/cpuset/foreground/tasks

service zygote_secondary /system/bin/app_process32 -Xzygote /system/bin --zygote --socket-
name=zygote_secondary
    class main
    socket zygote_secondary stream 660 root system
    onrestart restart zygote
    writepid /dev/cpuset/foreground/tasks
```

안드로이드 앱의 실행 원리

app_process란 무엇인가?

- /system/bin/app_process는 /system/bin/app_process64의 심볼릭 링크



POC SECURITY

Guess the workflow of Zygote

POC SECURITY

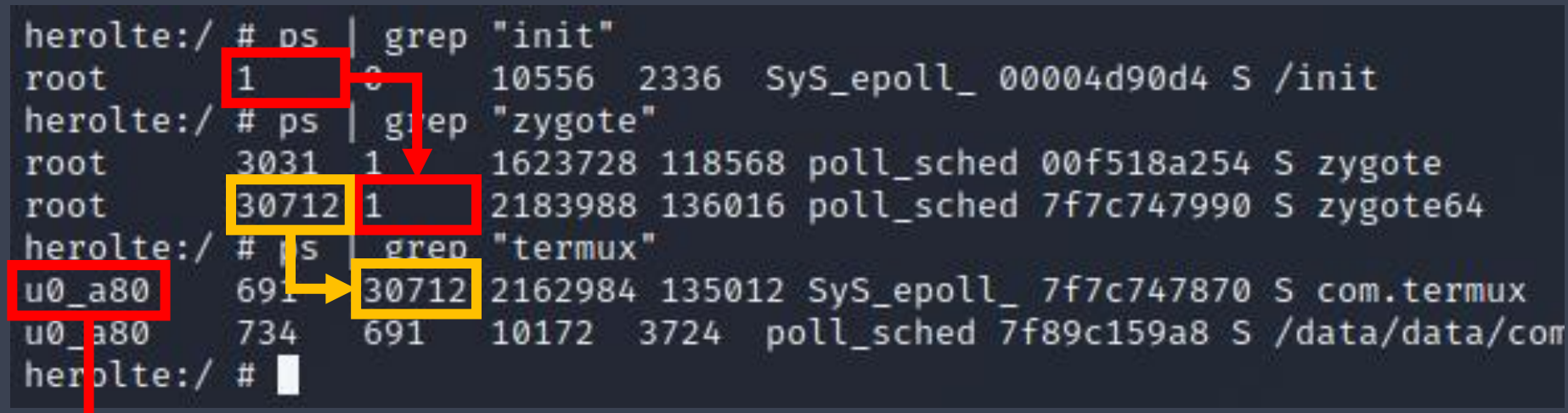
Zygote의 실행 흐름 예측

- Zygote는 Init의 의해 서비스 데몬으로 실행
- Zygote는 Init의 자식 프로세스로 Root 권한으로 실행
- Zygote의 자식 프로세스로 실행되는 안드로이드 애플리케이션도 **"원래"**는 **Root 권한으로 실행되었던 적이 있을 것**으로 예상

- **fork() 함수로 생성된 자식 프로세스는 기본적으로 부모 프로세스의 권한을 상속 받음**

POC SECURITY

Zygote의 실행 흐름 예측



Zygote가 애플리케이션(MainActivity)의 호출 전 자신의 권한을 낮춘 다는 것을 알 수 있음

POC SECURITY

Zygote source code analysis

Fact check

POC SECURITY

Zygote 소스코드 분석

- 안드로이드 7.1.2 Release 39 기준

[1] core/java/com/android/internal/os/ZygoteInit.java - main()
[2] core/java/com/android/internal/os/ZygoteInit.java - runSelectLoop()
[3] core/java/com/android/internal/os/ZygoteConnection.java - runOnce()
[4] core/java/com/android/internal/os/Zygote.java - forkAndSpecialize()
[5] core/jni/com_android_internal_os_Zygote.cpp - nativeForkAndSpecialize()
[6] core/jni/com_android_internal_os_Zygote.cpp - **ForkAndSpecializeCommon()**

- https://android.googlesource.com/platform/frameworks/base/+/refs/tags/android-7.1.2_r39/core/jni/com_android_internal_os_Zygote.cpp#444

POC SECURITY

Zygote 소스코드 분석

ForkAndSpecializeCommon


```
static pid_t ForkAndSpecializeCommon(JNIEnv* env, uid_t uid, gid_t gid, jintArray javaGids,
                                     jint debug_flags, jobjectArray javaRlimits,
                                     jlong permittedCapabilities, jlong effectiveCapabilities,
                                     jint mount_external,
                                     jstring java_se_info, jstring java_se_name,
                                     bool is_system_server, jintArray fdsToClose,
                                     jstring instructionSet, jstring dataDir)
```

POC SECURITY

Zygote 소스코드 분석

ForkAndSpecializeCommon


```
static pid_t ForkAndSpecializeCommon(JNIEnv* env, uid_t uid, gid_t gid, …) {
    …


    pid_t pid = fork();

    if (pid == 0) {
        // the client process
        …
    }
    else if (pid > 0) {
        // the parent process
        …
    }


    return pid;
}
```

Zygote 소스코드 분석

ForkAndSpecializeCommon


```
static pid_t ForkAndSpecializeCommon(JNIEnv* env, uid_t uid, gid_t gid, …) {
    …
    if (pid == 0) {
        …
        DropCapabilitiesBoundingSet(env);
        …
    }
    …
}
```

Zygote 소스코드 분석

ForkAndSpecializeCommon


```
static void DropCapabilitiesBoundingSet(JNIEnv* env) {
    for (int i = 0; prctl(PR_CAPBSET_READ, i, 0, 0, 0) >= 0; i++) {
        int rc = prctl(PR_CAPBSET_DROP, i, 0, 0, 0);
        if (rc == -1) {
            if (errno == EINVAL) {
                ALOGE("prctl(PR_CAPBSET_DROP) failed with EINVAL. Please verify "
                      "your kernel is compiled with file capabilities support");
            } else {
                RuntimeAbort(env, __LINE__, "prctl(PR_CAPBSET_DROP) failed");
            }
        }
    }
}
```

Zygote 소스코드 분석

ForkAndSpecializeCommon


```
static pid_t ForkAndSpecializeCommon(JNIEnv* env, uid_t uid, gid_t gid, …) {
    …
    if (pid == 0) {
        …
        DropCapabilitiesBoundingSet(env);
        …
        int rc = setresgid(gid, gid, gid);
        …
    }
    …
}
```

Zygote 소스코드 분석

ForkAndSpecializeCommon


```
static pid_t ForkAndSpecializeCommon(JNIEnv* env, uid_t uid, gid_t gid, …) {
    …
    if (pid == 0) {
        …
        DropCapabilitiesBoundingSet(env);
        …
        int rc = setresgid(gid, gid, gid);
        …
        rc = setresuid(uid, uid, uid);
        …
    }
    …
}
```

Zygote 소스코드 분석

ForkAndSpecializeCommon


```
static pid_t ForkAndSpecializeCommon(JNIEnv* env, uid_t uid, gid_t gid, …) {
    …
    if (pid == 0) {
        …
        DropCapabilitiesBoundingSet(env);
        …
        int rc = setresgid(gid, gid, gid);
        …
        rc = setresuid(uid, uid, uid);
        …
        SetCapabilities(env, permittedCapabilities, effectiveCapabilities);
        …
        env->CallStaticVoidMethod(gZygoteClass, gCallPostForkChildHooks, debug_flags,
                                  is_system_server, instructionSet);
        …
    }
    …
}
```

Zygote 소스코드 분석

ForkAndSpecializeCommon


```
static void SetCapabilities(JNIEnv* env, int64_t permitted, int64_t effective) {
    __user_cap_header_struct capheader;
    memset(&capheader, 0, sizeof(capheader));
    capheader.version = _LINUX_CAPABILITY_VERSION_3;
    capheader.pid = 0;

    __user_cap_data_struct capdata[2];
    memset(&capdata, 0, sizeof(capdata));
    capdata[0].effective = effective;
    capdata[1].effective = effective >> 32;
    capdata[0].permitted = permitted;
    capdata[1].permitted = permitted >> 32;

    if (capset(&capheader, &capdata[0]) == -1) {
      ALOGE("capset(%" PRId64 ", %" PRId64 ") failed", permitted, effective);
      RuntimeAbort(env, __LINE__, "capset failed");
    }
}
```

Zygote 소스코드 분석

ForkAndSpecializeCommon

ForkAndSpecializeCommon()

DropCapabilitiesBoundingSet	→ prctl	Drop all capabilities on bounding set
setresgid		Change process's gid
setresuid		Change process's uid
SetCapabilities	→ capset	Remove all capabilities

POC SECURITY

How to run application as root for Android

POC SECURITY

애플리케이션을 Root로 실행하는 방법

Zygote 프로세스 흐름 조작

- Zygote 프로세스는 모든 권한(Capability)을 Drop하고 UID와 GID를 변경
- 권한을 Drop하지 않고 UID와 GID를 변경하지 않도록 조작 시 Root로 실행 가능

- 안드로이드 애플리케이션을 Root로 실행하기 위해 모니터링 및 조작이 필요한 함수

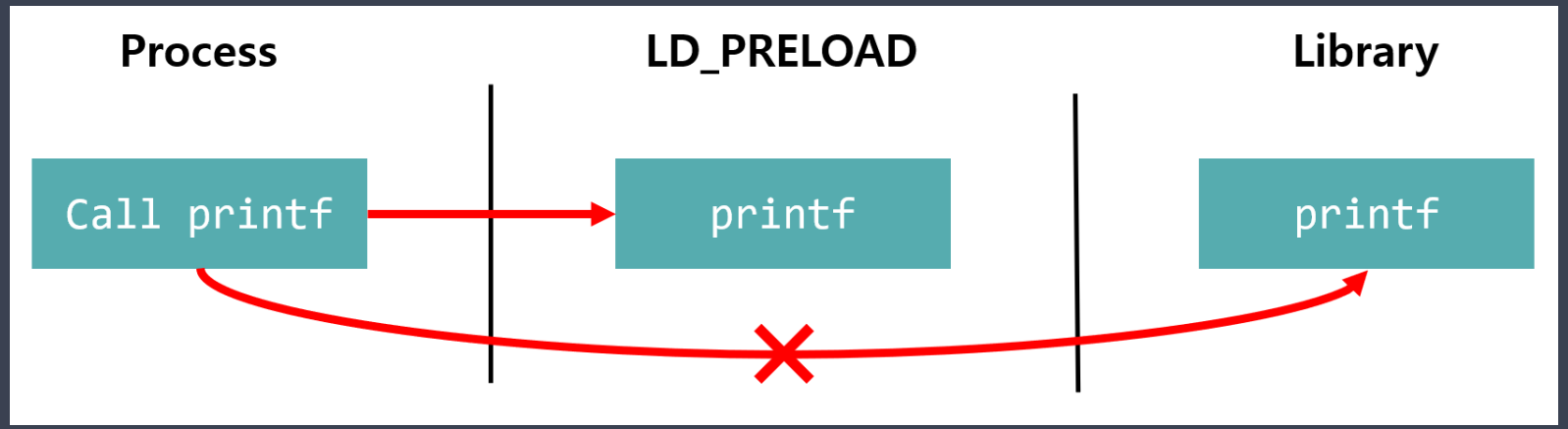
```
int setresuid(uid_t ruid, uid_t euid, uid_t suid);
int setresgid(gid_t rgid, gid_t egid, gid_t sgid);
int prctl(int option, ...);
int capset(cap_user_header_t hdrp, const cap_user_data_t datap);
```

POC SECURITY

애플리케이션을 Root로 실행하는 방법

LD_PRELOAD

- Zygote 프로세스의 특정 함수를 조작하기 위해 LD_PRELOAD 기술을 사용
- LD_PRELOAD는 Linux 운영체제에서 함수 후킹을 위해 자주 사용되는 기술



- 자세한 설명 : https://blog.crazyhacker.kr/140

POC SECURITY

애플리케이션을 Root로 실행하는 방법

Github

- 여기서부터 자세한 코드는 Github을 참고해주세요
- Link : https://github.com/jungjin0003/Run-Application-as-Root-for-Android

Run Application as Root for Android : RARA

A library that helps you run Android applications on root

Supported Android Versions

The Android versions that have confirmed the current behavior are as follows

Arch	Version
ARMv7a	7.1 (Nougat)
AArch64	7.1 (Nougat)

Getting Started

Please follow the following procedure to use RARA
Click here to download compiled files and if you're donw downloading it, proceed from here
To use automated applications on Android, click here to download APK

POC SECURITY

애플리케이션을 Root로 실행하는 방법

후킹 라이브러리 코드


```c
int prctl(int option, ...)
{
    va_list args;
    va_start(args, option);

    long arg2 = va_arg(args, long);
    long arg3 = va_arg(args, long);
    long arg4 = va_arg(args, long);
    long arg5 = va_arg(args, long);

    va_end(args);

    if (option == PR_CAPBSET_DROP)
        return 0;

    if (origin_prctl == NULL)
        origin_prctl = dlsym(RTLD_NEXT, "prctl");

    return origin_prctl(option, arg2, arg3, arg4, arg5);
}
```

애플리케이션을 Root로 실행하는 방법

후킹 라이브러리 코드


```
int setresgid(gid_t rgid, gid_t egid, gid_t sgid)
{
    if (isRootApplication(rgid))
        return 0;

    if (origin_setresgid == NULL)
        origin_setresgid = dlsym(RTLD_NEXT, "setresgid");

    return origin_setresgid(rgid, egid, sgid);
}
```

POC SECURITY

애플리케이션을 Root로 실행하는 방법

후킹 라이브러리 코드


```
int setresuid(uid_t ruid, uid_t euid, uid_t suid)
{
    if (isRootApplication(ruid))
    {
        pid = getpid();
        return 0;
    }

    dropCapabilitiesBoundingSet();

    if (origin_setresuid == NULL)
        origin_setresuid = dlsym(RTLD_NEXT, "setresuid");

    return origin_setresuid(ruid, euid, suid);
}
```

POC SECURITY

애플리케이션을 Root로 실행하는 방법

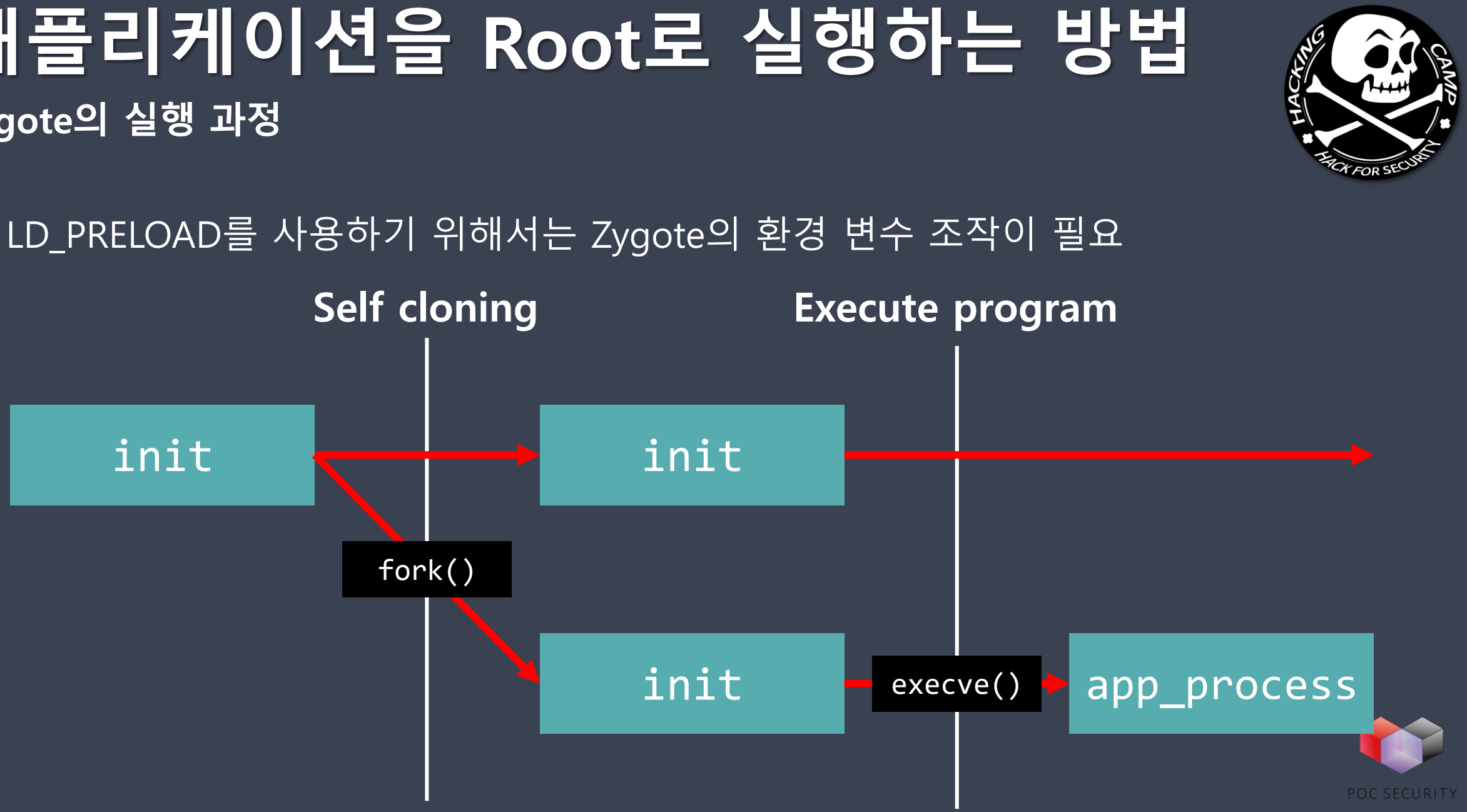
후킹 라이브러리 코드


```
int capset(cap_user_header_t __hdr_ptr, const cap_user_data_t __data_ptr)
{
    if (pid == getpid())
        return 0;

    if (origin_capset == NULL)
        origin_capset = dlsym(RTLD_NEXT, "capset");

    return origin_capset(__hdr_ptr, __data_ptr);
}
```

POC SECURITY

애플리케이션을 Root로 실행하는 방법

Zygote의 실행 과정



- LD_PRELOAD를 사용하기 위해서는 Zygote의 환경 변수 조작이 필요

POC SECURITY

애플리케이션을 Root로 실행하는 방법

Zygote의 실행 과정

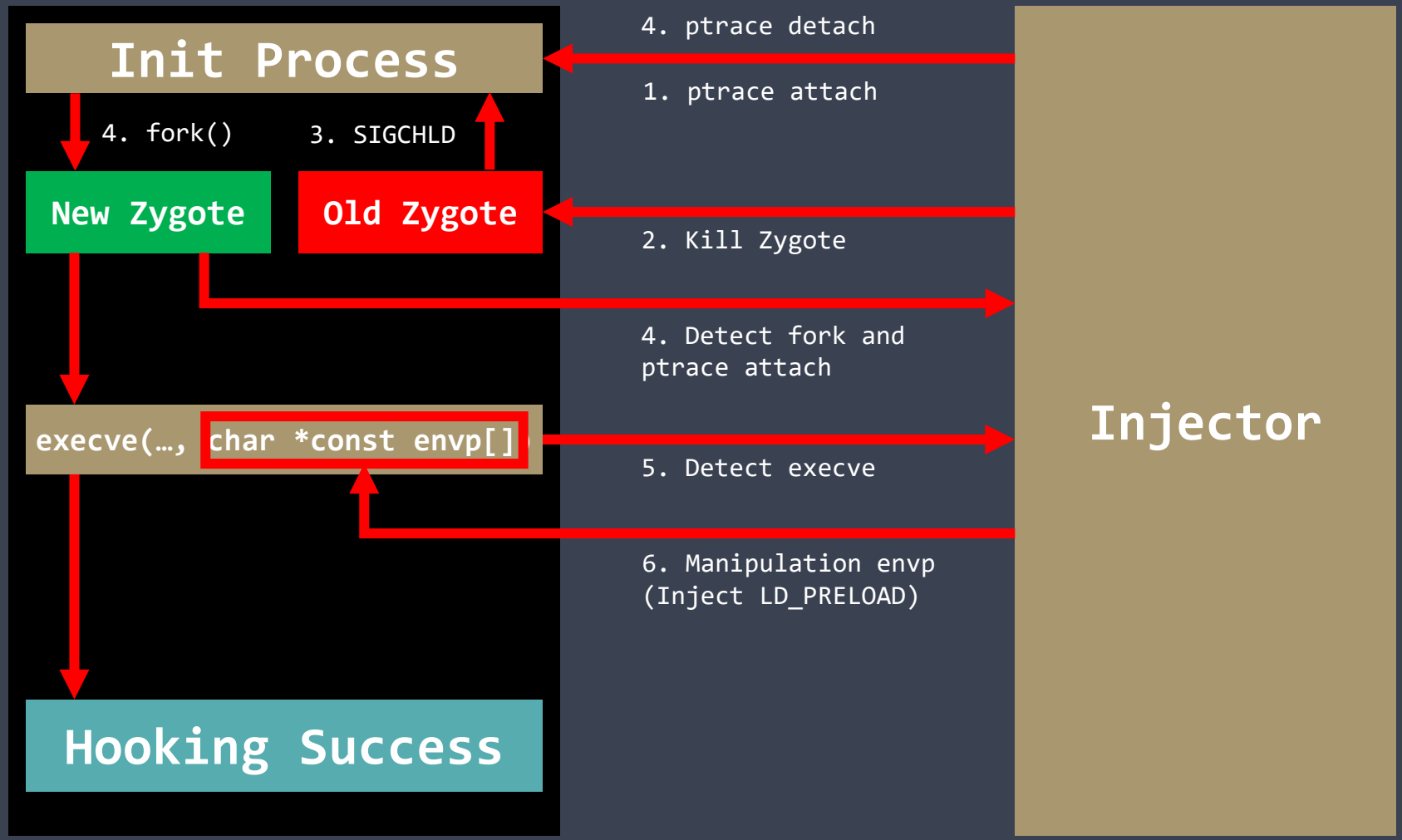
- execve() 함수에는 envp 인자가 존재, 해당 인자가 프로그램의 환경 변수를 설정
- fork()된 init 프로세스가 execve() 함수를 호출할 때 LD_PRELOAD를 삽입 또는 조작

```
int execve(const char *pathname, char *const argv[], char *const envp[]);
```

POC SECURITY

애플리케이션을 Root로 실행하는 방법

Zygote 프로세스에 LD_PRELOAD 환경 변수를 설정하는 방법



POC SECURITY

감사합니다

2024.08.18

POC SECURITY