

Beginning of **Android Kernel** analysis

2024. 8. 31

78ResearchLab

이소연 연구원

C O N T E N T S

01

Preparation

01 Installation

02 Initialization

02

Build

01 AOSP Build

02 Custom Kernel
Build

03





Debugging

01 Emulating

02 Debugging

0. Detailed table of Contents

Detailed table of Contents

Preparation	Build		Debugging
1. Installation & Initialization	2. AOSP Build	3. Custom Kernel Build	4. Emulating & Debugging
 <p>1.1 Installation for Environment setting</p> <p>1.2 Initializing the AOSP tree</p> <p>1.3 AOSP manifest structure</p>	 <p>2.1 OpenSource Project (Each Vender)</p> <p>2.2 Vendor's Example</p> <p>2.3 devices.xml 파일 예시를 통한 manifest 이해</p> <p>2.4 Google에서 기본적으로 AOSP 빌드를 제공하는 device</p> <p>2.5 AOSP Build 진행</p>	 <p>3.1 Vendor's Kernel OpenSource 관리 방법</p> <p>3.2 Cross Compile Toolchain setting</p> <p>3.3 Build Config & Variable Setting</p> <p>3.4 Custom Kernel Build 진행 & Clean up</p>	 <p>4.1 About Android Emulator</p> <p>4.2 Creating AVD</p> <p>4.3 Basic example of emulating Android</p> <p>4.4 (option) KASAN 컴파일 옵션 적용 후 커널 빌드</p>

Preparation

01 Installation

02 Initialization

1. Installation

Installation for Environment setting

JAVA (optional)

```
$ sudo apt-get purge openjdk-* icedtea-* icedtea6-*  
$ sudo apt-get update  
$ sudo apt-get install openjdk-11-jdk  
  
$ java -version  
openjdk version "11.0.20.1" 2023-08-24
```

- Android를 빌드하려면 벤더사 별로 특정 Java version 이 필요
- 기존에 설치된 JAVA 를 제거 후, 특정 버전으로 재설치하는 명령어

Necessary Packages

```
$ sudo apt-get install bison g++-multilib git \  
gperf libxml2-utils make zlib1g-dev:i386 zip \  
liblz4-tool libncurses5 libssl-dev bc flex curl \  
python-is-python3 zlib1g-dev libelf-dev dwarves
```

- Source Code 를 android Device에서 실행할 수 있는 바이너리로 컴파일하기 위해 필요한 tools , 소프트웨어 패키지 및 라이브러리 세트설치

1. Installation

Installation for Environment setting

Repo tool

- Android AOSP는 다양한 Git Repository로 관리되는 프로젝트들을 포함, 이를 일괄적으로 관리하기 위한 tool
- Google에서 제공하는 Repo tool 설치, 권한 및 환경변수 설정
- 자세한 repo tool 명령어는 Android 공식홈페이지(source.android.com)의 문서에서 확인 가능

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo

$ nano ~/.bashrc

$ export PATH=~/bin:$PATH

$ source ~/.bashrc
```

2. Initialization

Initializing the AOSP tree

Repo init

- Android Source tree를 초기 설정할 때, repo init 명령어를 사용
- AOSP는 Android 릴리즈 버전, 보안 패치 버전 별로 기본적인 manifest를 제공
- repo init 명령을 통해 희망하는 버전의 브랜치를 받아와 이를 기반으로 추가 설정 및 빌드 진행
- Google Opensource(android.googlesource.com/platform/manifest/+refs)에서 branch 목록 확인 가능

```
$ mkdir ~/android
$ cd ~/android
$ repo init -u https://android.googlesource.com/platform/manifest -b <branch>
```

```
<branch list>
main
...
android-12.0.0_r7
...
android-13.0.0_r78
android-13.0.0_r79
...
android-14.0.0_r55
...
```

2. Initialization

AOSP manifest structure

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
  <!-- Project metadata -->
  <remote name="aosp"
    fetch=".."
    review="https://android-review.googlesource.com/" />
  <default revision="refs/tags/android-13.0.0_r75"
    remote="aosp"
    sync-j="4" />

  <!-- Build utilities -->
  <project path="build/make" name="platform/build" groups="pdk" > ... </project>
  <project path="build/bazel" name="platform/build/bazel" groups="pdk" > ... </project>
  <project path="external/lzma" name="platform/external/lzma" groups="pdk" />

  <!-- Device, hardware components -->
  <project path="device/common" name="device/common" groups="pdk-cw-fs,pdk" />
  <project path="hardware/qcom/wlan" name="platform/hardware/qcom/wlan" groups="qcom_wlan,pdk-qcom" />

  <!-- Prebuilts binaries -->
  <project path="prebuilts/jdk/jdk11" name="platform/prebuilts/jdk/jdk11" groups="pdk" clone-depth="1" />
</manifest>
```


Build

01 AOSP Build

02 Custom Kernel Build

1. AOSP Build

Google에서 기본적으로 AOSP 빌드를 지원하는 device

android.googlesource.com/device/

- Google 오픈소스 프로젝트에서 기본적으로 AOSP 빌드를 할 수 있도록 제공하는 디바이스들이 존재
- android.googlesource.com/device/ 경로에 벤더 및 Codename 별로 Code 관리

Google Git

[Code Review](#)

[android](#) / **device**

Name	samsung_slsi/arndale
aaeon/upboard	samsung/crespo
amlogic/yukawa	samsung/crespo4g
amlogic/yukawa-kernel	samsung/maguro
asus/deb	samsung/manta
asus/flo	samsung/toro
asus/flo-kernel	samsung/toroplus
asus/fugu	samsung/torospr
asus/fugu-kernel	samsung/tuna

1. AOSP Build

AOSP Build 진행

Build setup

- 빌드 변수 설정, 유틸리티 명령어 정의 등 빌드 시스템 초기화
- 앞서 빌드 설정 파일들을 통해 생성한 빌드 타겟 확인, 지정
- 메뉴에 모든 선택 항목이 포함되어 있지는 않음, 해당 경우 매개변수로 입력



```
$ source build/envsetup.sh && lunch
```

```
1. aosp_codename-eng
2. aosp_codename-userdebug
```

```
Which would you like? [aosp_codename-eng]
Pick from common choices above (e.g. 13)
or specify your own (e.g. aosp_codename-eng):
```



Build

- make, m, mm, mmm 등의 명령을 통해 빌드
- 빌드 완료 시 out 디렉터리에 이미지 파일 생성



```
$ make -j$(nproc)
```

```
out/
├─ boot.img
├─ product.img
├─ system.img
├─ ...
└─ vendor.img
```

2. Custom Kernel Build

특정 Vendor's Kernel OpenSource 관리 방법 예시

MiCode / Xiaomi_Kernel_OpenSource Public

<> Code Issues 5k+ Pull requests 34 Actions Projects Wiki Security Insights

README 210 Branches 0 Tags

gengbo Update README.md 5a1f974 · last month 230 Commits

README.md Update README.md last month

README

venus-r-oss	Mi 11	Android R	LA.UM.9.14.r1-10000-LAHAINA.0
veux-r-oss	Redmi Note 11E Pro, Redmi Note 11 Pro 5G, Redmi Note 11 Pro+ 5G	Android R	LA.UM.9.16.r1-07900-MANNAR.0-1
vili-r-oss	Xiaomi 11T Pro	Android R	LA.UM.9.14.r1-16700-LAHAINA.0

MiCode / Xiaomi_Kernel_OpenSource Public

<> Code Issues 5k+ Pull requests 34 Actions

Branches

Overview Active Stale All

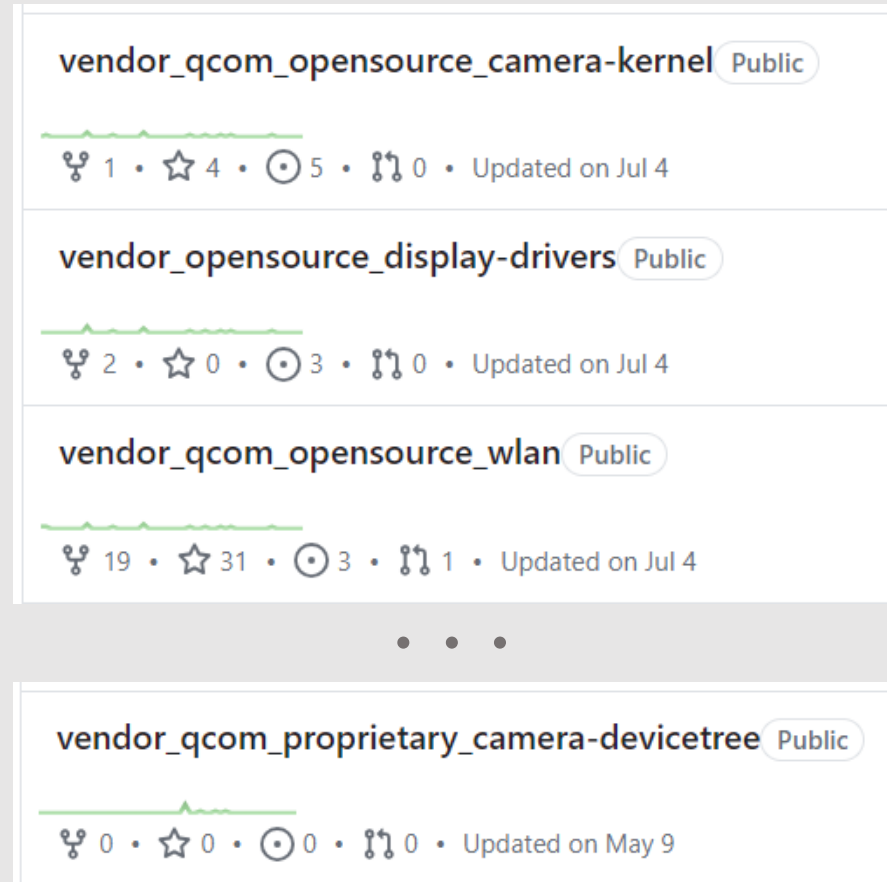
Search branches...

Branch

- README
- moon-u-oss
- breeze-u-oss
- chenfeng-u-oss
- neridot-u-oss

2. Custom Kernel Build

특정 Vendor's Kernel OpenSource 관리 방법 예시



Xiaomi OpenSource

Kernel

- 해당 예시 vendor사의 경우, 하나의 레포지토리에서 각 디바이스별 코드 네임을 Branch로 커널을 관리
- 빌드를 원하는 디바이스의 Branch로 이동하여 빌드 진행

Different Vendor's driver Code

- 해당 Vendor사에서 작성하지 않은 외부 Vendor 사의 코드의 경우 별도의 레포지토리를 이용해 관리
- 그림의 예시는 Xioami에서 따로 관리하는 qcom사의 드라이버 코드

2. Custom Kernel Build

Install Cross Compile Toolchain

Install Toolchain

- Arm64 아키텍처에 대한 Build 를 리눅스 환경에서 진행시키므로, 다른 아키텍처 환경에 대한 Compiler가 필요
- Google 공식에서 제공하는 android.googlesource.com이 해당 Cross Compile Toolchain을 포함
- (option) Kernel Build 시, android 13 이상부터는 공식적으로 gcc를 지원하지 않으므로 clang 사용이 필요



```
$ git clone https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/aarch64/aarch64-  
linux-android-4.9 -b android11-release
```

```
(option) $ sudo apt install clang
```

2. Custom Kernel Build

Build Environment Variable Setting

Build variable initialization

- 빌드 타겟 시스템 아키텍처와 컴파일러 설정
- 크로스 컴파일 툴체인은 Google Opensource의 prebuilt 바이너리나 NDK에 포함된 바이너리 등을 사용
- 빌드하려는 안드로이드 버전에 따라 GCC/Clang 사용 결정

```
$ export ARCH=arm64
$ export CROSS_COMPILE=~/.android/aarch64-linux-android-4.9/bin/aarch64-linux-android

$ export CC=clang
$ export CLANG_TRIPLE=aarch64-linux-gnu-
```

2. Custom Kernel Build

Build Configuration Files Setting

Build Configuration Files

- 벤더측은 안드로이드 기기에 적용되는 하드웨어에 맞게 커널 컴파일 설정을 저장하여 배포
- Default Config(_defconfig), Generic Kernel Image(_GKI), Debug symbol(_debug) 등 파일 이름의 구분자를 잘 확인하여 용도에 맞는 설정 파일 선택



```
~/android/Xiaomi_Kernel_OpenSource$ ls arch/arm64/configs/vendor/veux  
veux_debug.config  veux_GKI.config  veux_QGKI.config  veux-qgki-debug_defconfig
```


2. Custom Kernel Build

Custom Kernel Config setup

.config - Linux/arm64 5.4.134 Kernel Configuration

Linux/arm64 5.4.134 Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenu if there are none).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> module if available, < > module capable.
Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [] built-in [] excluded <M> module < > module capable

General setup --->

- [] Enable or Disable Zone DMA
- Platform selection --->
- Kernel Features --->
- Boot options --->
- Kernel compression method (Build uncompressed kernel image) --->
- [] Boot information from bootloader
- [] IS QGKI build
- [] enable DT overlay compilation support
- [] enable handler for TLB conflict
- Power management options --->
- CPU Power Management --->
- Firmware Drivers --->
- [] ACPI (Advanced Configuration and Power Interface) Support ----
- [] Virtualization ----
- [*] ARM64 Accelerated Cryptographic Algorithms --->
- General architecture-dependent options --->
- [*] Enable loadable module support --->

v(+)

<Select>

<Exit>

<Help>

<Save>

<Load>

Config setup

- 벤더측에서 제공하는 빌드 설정 파일을 토대로 커스텀 설정
- 부트 옵션, 펌웨어 드라이버 적재 등 빌드 목적에 맞는 옵션 선택



```
$ make -j$(nproc) 0=out vendor/veux-qgki-debug_defconfig  
$ make -j$(nproc) 0=out menuconfig
```

2. Custom Kernel Build

Kernel Makefile manual



```
$ make help
```

```
Cleaning targets:
```

```
clean      - Remove most generated files but keep the config and
              enough build support to build external modules
mrproper   - Remove all generated files + config + various backup files
...
```

```
Configuration targets:
```

```
config     - Update current config utilising a line-oriented program
menuconfig - Update current config utilising a menu based program
...
```

```
Other generic targets:
```

```
all        - Build all targets marked with [*]
* vmlinux   - Build the bare kernel
* modules  - Build all modules
...
```

```
Devicetree:
```

```
* dtbs      - Build device tree blobs for enabled boards
dtbs_install - Install dtbs to /boot/dtbs/
...
```

```
Architecture specific targets (arm64):
```

```
* Image.gz   - Compressed kernel image (arch/arm64/boot/Image.gz)
Image        - Uncompressed kernel image (arch/arm64/boot/Image)
...
```

2. Custom Kernel Build

Custom Kernel Build 진행 & Clean up



빌드

```
$ make -j$(nproc) 0=out
```

초기화

```
$ make -j$(nproc) 0=out clean
```

```
$ make -j$(nproc) 0=out mrproper
```

```
ye0n@ubuntu:~/Desktop/Xiaomi_Kernel_OpenSource$ make -j$(nproc) 0=out
make[1]: Entering directory '/home/ye0n/Desktop/Xiaomi_Kernel_OpenSource/out'
arch/arm64/Makefile:52: Detected assembler with broken .inst; disassembly will
/home/ye0n/Desktop/Xiaomi_Kernel_OpenSource/Makefile:1039: ###enable user root
GEN      Makefile
WRAP     arch/arm64/include/generated/uapi/asm/kvm_para.h
WRAP     arch/arm64/include/generated/uapi/asm/errno.h
WRAP     arch/arm64/include/generated/uapi/asm/ioctl.h
WRAP     arch/arm64/include/generated/uapi/asm/ioctls.h
WRAP     arch/arm64/include/generated/uapi/asm/ipcbuf.h
WRAP     arch/arm64/include/generated/uapi/asm/mman.h
WRAP     arch/arm64/include/generated/uapi/asm/msgbuf.h
HOSTCC   scripts/dtc/dtc.o
WRAP     arch/arm64/include/generated/uapi/asm/poll.h
WRAP     arch/arm64/include/generated/uapi/asm/resource.h
WRAP     arch/arm64/include/generated/uapi/asm/sembuf.h
WRAP     arch/arm64/include/generated/uapi/asm/shmbuf.h
```

Emulating & Debugging

#. Emulating & Debugging

About Android Emulator

Emulate



When?

- Crash Dump 분석
- Kernel / App 퍼징
- App 개발 및 테스트

Install Android Emulator



Android Studio를 통해
Android SDK 구성요소 설치



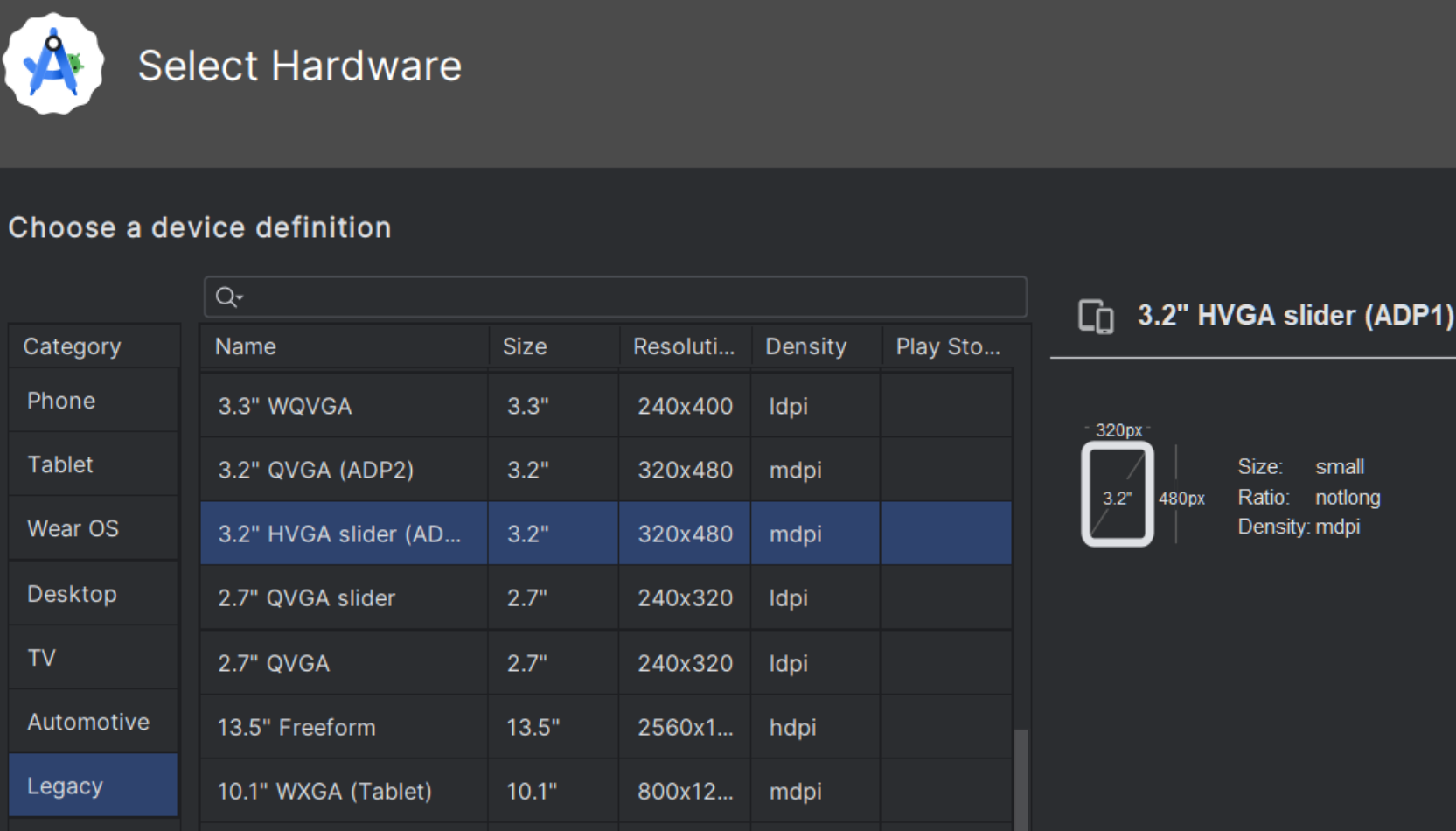
SDK의 Android Emulator로
안드로이드 가상 기기(AVD) 생성



Emulating 진행

#. Emulating & Debugging

Creating AVD



Select Hardware

Choose a device definition

Category	Name	Size	Resoluti...	Density	Play Sto...
Phone	3.3" WQVGA	3.3"	240x400	ldpi	
Tablet	3.2" QVGA (ADP2)	3.2"	320x480	mdpi	
Wear OS	3.2" HVGA slider (AD...	3.2"	320x480	mdpi	
Desktop	2.7" QVGA slider	2.7"	240x320	ldpi	
TV	2.7" QVGA	2.7"	240x320	ldpi	
Automotive	13.5" Freeform	13.5"	2560x1...	hdpi	
Legacy	10.1" WXGA (Tablet)	10.1"	800x12...	mdpi	

3.2" HVGA slider (ADP1)

320px
3.2"
480px

Size: small
Ratio: notlong
Density: mdpi

#. Emulating & Debugging

Creating AVD

Select a system image

Recommended

x86 Images

Other Images

Release Name	API Level ▾	ABI	Target
UpsideDownCake	34	x86_64	Android 14.0 (Google APIs)
Tiramisu ▾	33	x86_64	Android 13.0 (Google APIs)
Sv2 ▾	32	x86_64	Android 12L (Google APIs)
S ▾	31	x86_64	Android 12.0 (Google APIs)
R ▾	30	x86	Android 11.0 (Google APIs)
Q ▾	29	x86	Android 10.0 (Google APIs)
Pie ▾	28	x86	Android 9.0 (Google APIs)
Oreo ▾	27	x86	Android 8.1 (Google APIs)
Oreo ▾	26	x86	Android 8.0 (Google APIs)

#. Emulating & Debugging

Basic example of emulating Android

```
$ git clone https://github.com/cloudfuzz/android-kernel-exploitation ~/workshop

$ cd workshop/android-4.14-dev/
$ repo init --depth=1 -u https://android.googlesource.com/kernel/manifest \
  -b q-goldfish-android-goldfish-4.14-dev

$ cp ../custom-manifest/default.xml .repo/manifests/
$ repo sync -c --no-tags --no-clone-bundle -j`nproc` -qemu -s -S

$ BUILD_CONFIG=../build-configs/goldfish.x86_64.kasan build/build.sh

$ emulator -show-kernel -no-snapshot -wipe-data -avd CVE-2019-2215 \
  -kernel /home/ubuntu/workshop/android-4.14-dev/out/kasan/dist/bzImage \
  -qemu -s -S
```

```
$ tree -F -L 2 workshop
workshop/
├── android-4.14-dev/
├── build-configs/
│   ├── goldfish.x86_64.kasan
│   └── goldfish.x86_64.relwithdebinfo
├── custom-manifest/
│   └── default.xml
├── Dockerfile
├── exploit/
│   ├── CMakeLists.txt
│   ├── common.h
│   ├── exploit.cpp
│   ├── exploit.h
│   ├── Makefile
│   └── trigger.cpp
├── gdb/
│   ├── dynamic-analysis.py
│   └── root-me.py
├── gitbook/
│   ├── book.json
│   ├── chapters/
│   ├── images/
│   ├── README.md
│   └── SUMMARY.md
├── LICENSE
├── patch/
│   └── cve-2019-2215.patch
```


#. Emulating & Debugging

(option) KASAN 컴파일 옵션 적용 후 커널 빌드

Apply KASAN Option

- KASAN 및 KCOV는 일반 빌드에선 사용되지 않는 컴파일 플래그
- Config file 에서 압축 옵션 비활성화 및 Sanitization , Coverage 관련 옵션은 활성화
- 수정된 Config file 을 적용하여 다시 빌드 진행

```
$ cd arch/arm64/configs
$ cp veu-xgki-debug_defconfig veu-xasan_defconfig
$ vim veu-xasan_defconfig

. . .
<!-- CONFIG_KERNEL_LZ4=y -->
CONFIG_KASAN=y
CONFIG_KASAN_INLINE=y
CONFIG_KCOV=y
CONFIG_SLUB=y
CONFIG_SLUB_DEBUG=y
. . .
```

#. Emulating & Debugging

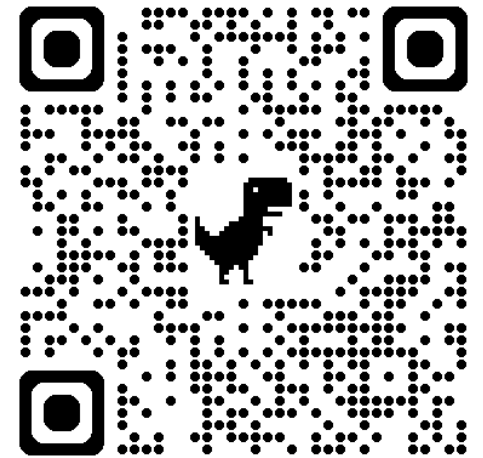
Beginner recommended example

CVE-2019-2215

- 취약점에 대한 Reference 다수 존재
- Poc 코드 및 exploit 코드가 github에 공개
- 해당 취약한 커널 환경 구축 자료 및 코드 존재

Q n A

감사합니다



2024. 8. 31