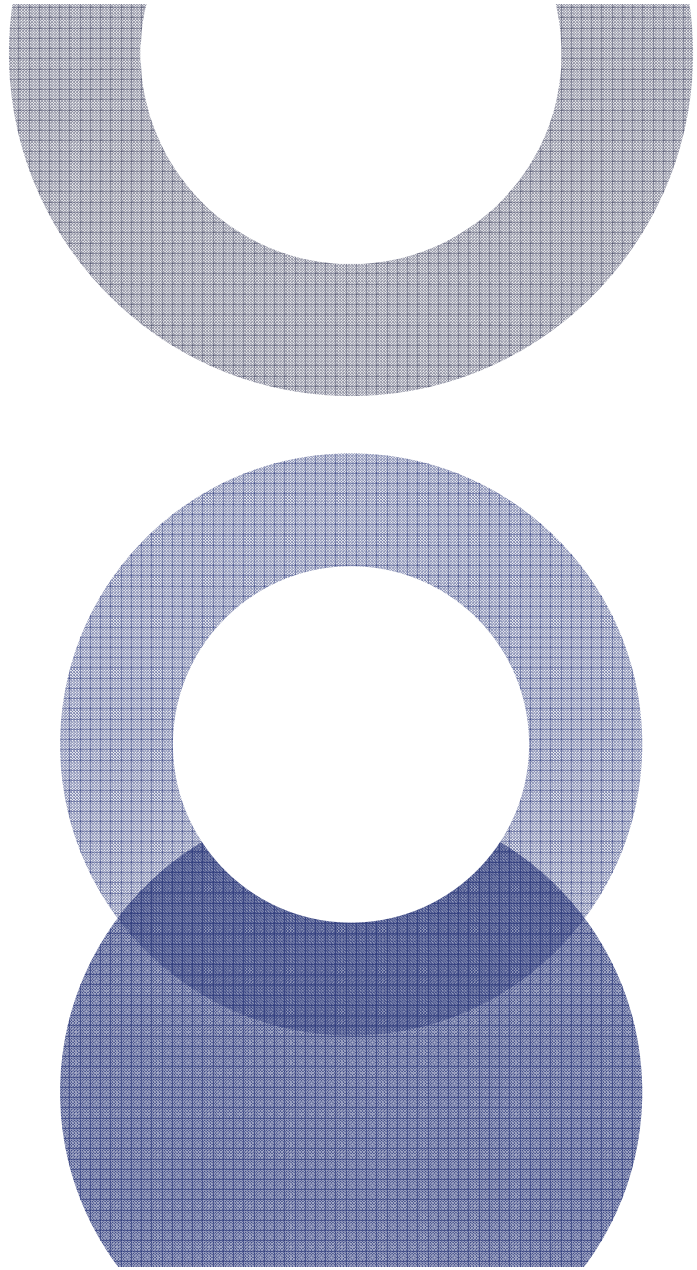
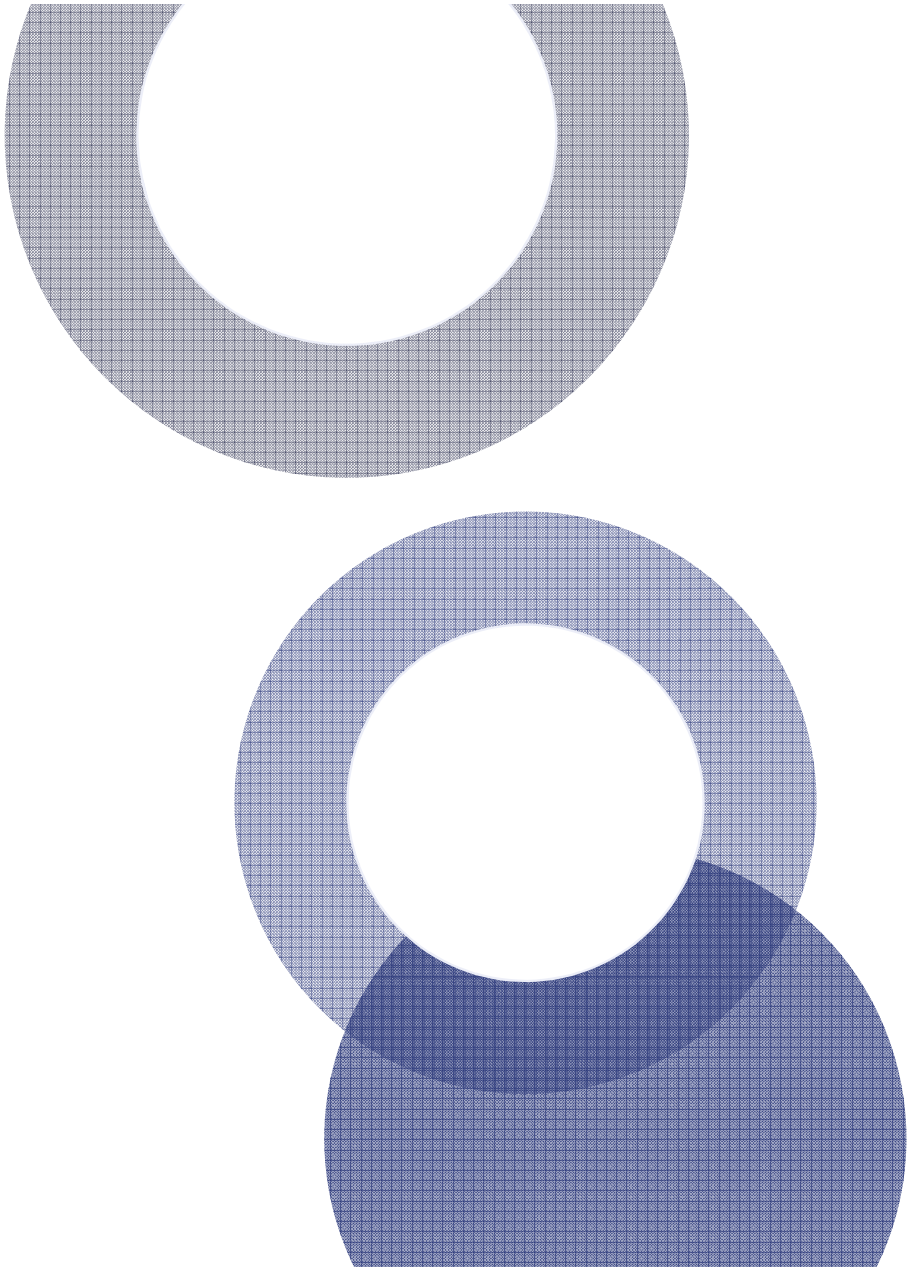
안드로이드 악성 앱 분석을 위한

언패커 개발

2022.11.01
서동훈
전유민
강민영
정재훈



Contents



I

프로젝트 소개

1. 팀원 소개
2. 프로젝트 배경
3. 프로젝트 개요
4. 추진 경과

팀원 소개

저희 팀원을 소개합니다!

서 동 훈

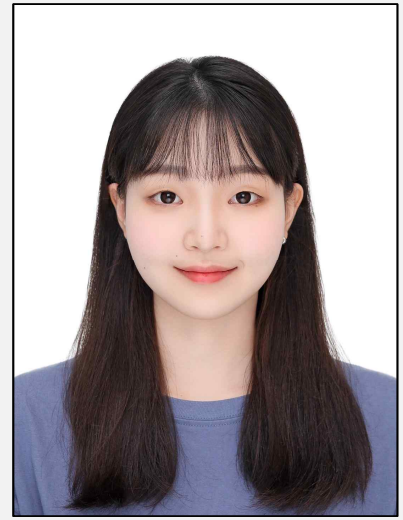
총괄
악성 앱 분석 및 개발

전 유 민

언패커 분석 및 Yara 개발

정 재 훈

언패커 분석 및 웹 개발



강 민 영

언패커 분석 및 웹 개발

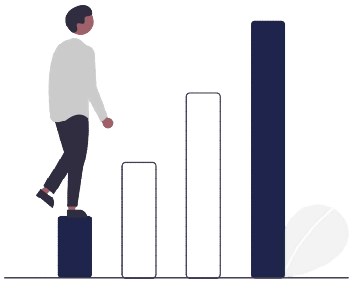
I

프로젝트 소개 – 배경

프로젝트 배경

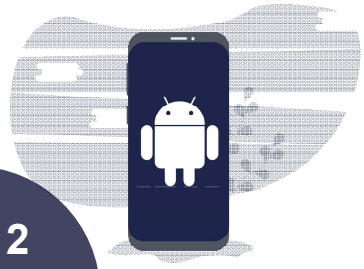
프로젝트 추진 배경 및 필요성과 목적

프로젝트 추진 배경 및 필요성



스마트폰 사용량 급증에 따른 피해 증가

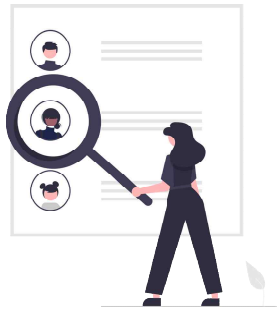
스마트폰 사용량이 지속적으로 증가함에 따라 이에 따른 피해도 점차 늘어나고 있습니다.



공격자의 안드로이드 악성 **APK** 파일 전파

공격자는 피싱, 스미싱, 몸캠 피싱 등과 같은 공격을 주로 안드로이드 악성 **APK** 파일을 이용하여 공격합니다.

1 2 3 목적



안드로이드 악성 **APK** 파일 탐지 연구 불충분

이와 관련된 도구나 연구가 국내에서는 활발히 이루어지지 않고 있습니다.

안드로이드 언패커 제작

이러한 문제를 해결하기 위해 안드로이드 언패커 개발을 진행합니다.

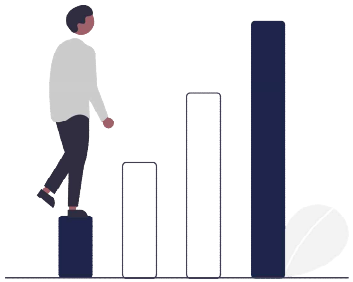
I

프로젝트 소개 – 배경

프로젝트 배경

✓ 프로젝트 추진 배경 및 필요성과 목적

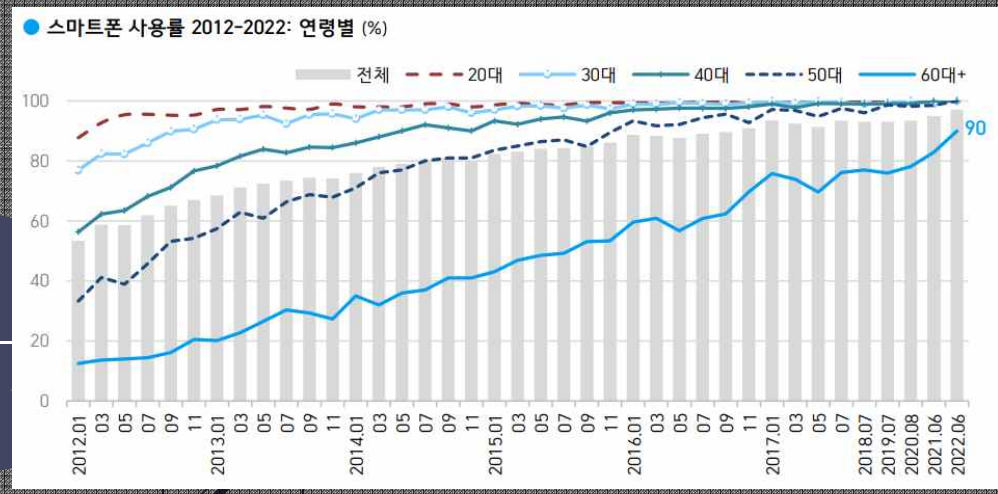
프로젝트 추진 배경 및 필요성



스마트폰 사용량 급증에 따른 피해 증가

스마트폰 사용량이 지속적으로 증가함에 따라 이에 따른 피해도 점차 늘어나고 있습니다.

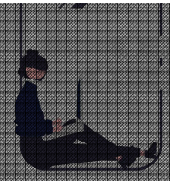
1



▲ 스마트폰 사용률

연도	피싱	스미싱	몸캠피싱
2017	545	667	1234
2018	1978	293	1406
2019	2874	207	1824
2020	1519	822	2583

▲ 피싱, 스미싱, 몸캠피싱 사건 증가량



이러한 문제를 해결하기 위해 안드로이드 언패커 개발을 진행합니다.

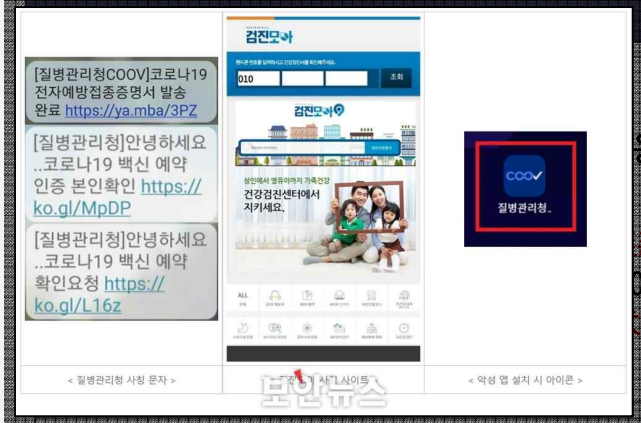
I

프로젝트 소개 – 배경

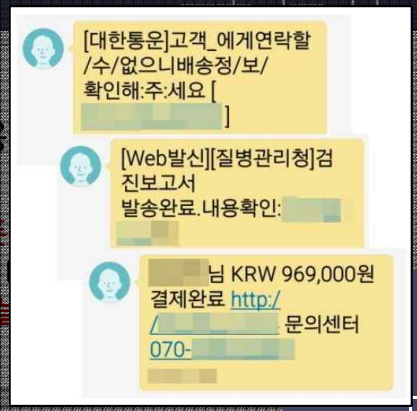
프로젝트 배경

프로젝트 추진 배경 및 필요성과 목적

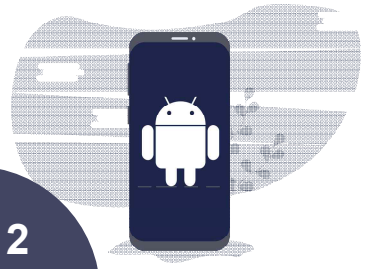
프로젝트 추진 배경 및 필요성



▲ 질병관리청 사칭



▲ 택배 사칭



공격자의 안드로이드 악성 **APK** 파일 전파

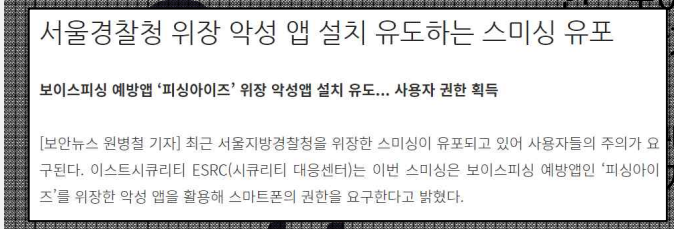
공격자는 피싱, 스미싱, 몸캠 피싱 등과 같은 공격을 주로 안드로이드 악성 **APK** 파일을 이용하여 공격합니다.

2

3 목적

안드로이드 언패커 제작

이러한 문제를 해결하기 위해 안드로이드 언패커 개발을 진행합니다.

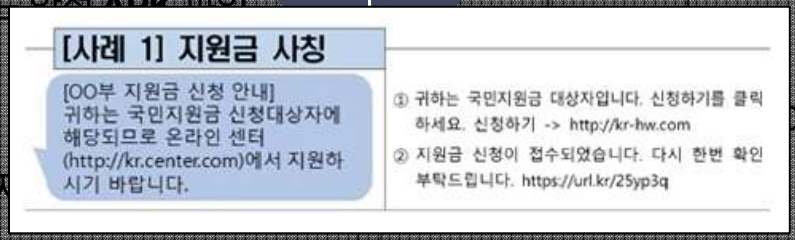


서울경찰청 위장 악성 앱 설치 유도하는 스미싱 유포

보이스피싱 예방앱 '피싱아이즈' 위장 악성앱 설치 유도... 사용자 권한 획득

[보안뉴스 원병철 기자] 최근 서울지방경찰청을 위장한 스미싱이 유포되고 있어 사용자들의 주의가 요구된다. 이스트시큐리티 ESRC(시큐리티 대응센터)는 이번 스미싱은 보이스피싱 예방앱인 '피싱아이즈'를 위장한 악성 앱을 활용해 스마트폰의 권한을 요구한다고 밝혔다.

▲ 경찰청 위장 앱



▲ 지원금 사칭

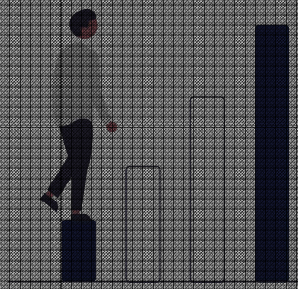
I

프로젝트 소개 – 배경

프로젝트 배경

✓ 프로젝트 추진 배경 및 필요성과 목적

프로젝트 추진 배경 및 필요성



스마트폰

검색결과 총 0건
전체=안드로이드 언패커

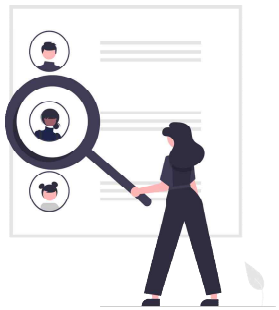
- 검색어의 철자가 정확한지 확인해 주세요.
- 보다 일반적인 단어로 검색해 보세요.
- 유사한 뜻을 가진 다른 단어로 검색해 보세요.
- 검색 방법에 관한 안내는 FAQ를 참고해 주세요.
- 만약 계속해서 검색이 되지 않으실 경우 고객센터로 문의 바랍니다.

검색결과 총 0건
전체=안드로이드 언패킹

- 검색어의 철자가 정확한지 확인해 주세요.
- 보다 일반적인 단어로 검색해 보세요.
- 유사한 뜻을 가진 다른 단어로 검색해 보세요.
- 검색 방법에 관한 안내는 FAQ를 참고해 주세요.
- 만약 계속해서 검색이 되지 않으실 경우 고객센터로 문의 바랍니다.

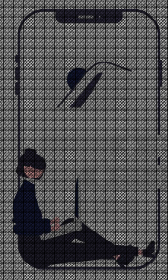
▲ DBpia 검색결과

3 목적



안드로이드 악성 **APK** 파일
탐지 연구 불충분

이와 관련된 도구나 연구가 국내에서는 활발히 이루어지지 않고 있습니다.



안드로이드 언패커 제작

이러한 문제를 해결하기 위해 안드로이드 언패커 개발을 진행합니다.

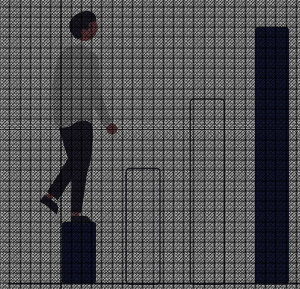
I

프로젝트 소개 – 배경

프로젝트 배경

✓ 프로젝트 추진 배경 및 필요성과 목적

프로젝트 추진 배경 및 필요성



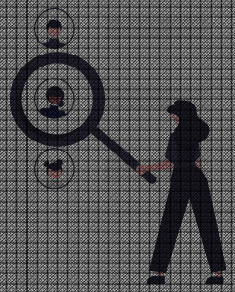
스마트폰 사용량 급증에 따른 피해 증가

스마트폰 사용량이 지속적으로 증가함에 따라 이에 따른 피해도 점차 늘어나고 있습니다.

공격자의 안드로이드 악성 APK 파일 전파

공격자는 피싱, 스미싱, 몸캠 피싱 등과 같은 공격을 주로 안드로이드 악성 APK 파일을 이용하여 공격합니다.

1 2 3 목적



안드로이드 악성 APK 파일 탐지 연구 불충분

이와 관련된 도구나 연구가 국내에서는 활발히 이루어지지 않고 있습니다.

안드로이드 언패커 제작

이러한 문제를 해결하기 위해 안드로이드 언패커 개발을 진행합니다.

I

프로젝트 소개 – 개요

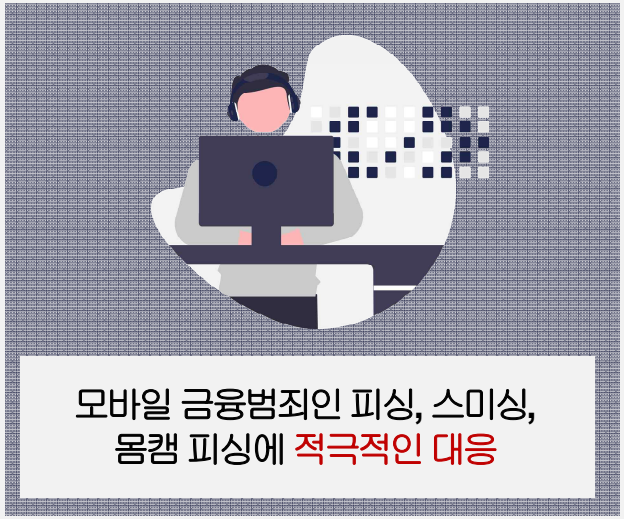
프로젝트 개요

안드로이드 언패커 제작 개요

안드로이드 언패커 제작

분석가의 실무 능력 향상

국내에서 활발히 이루어지지 않는 패킹 기법에 관한 연구 및 방법론 제시



모바일 금융범죄인 피싱, 스미싱, 몸캠 피싱에 적극적인 대응

I

프로젝트 소개 – 개요

추진 경과

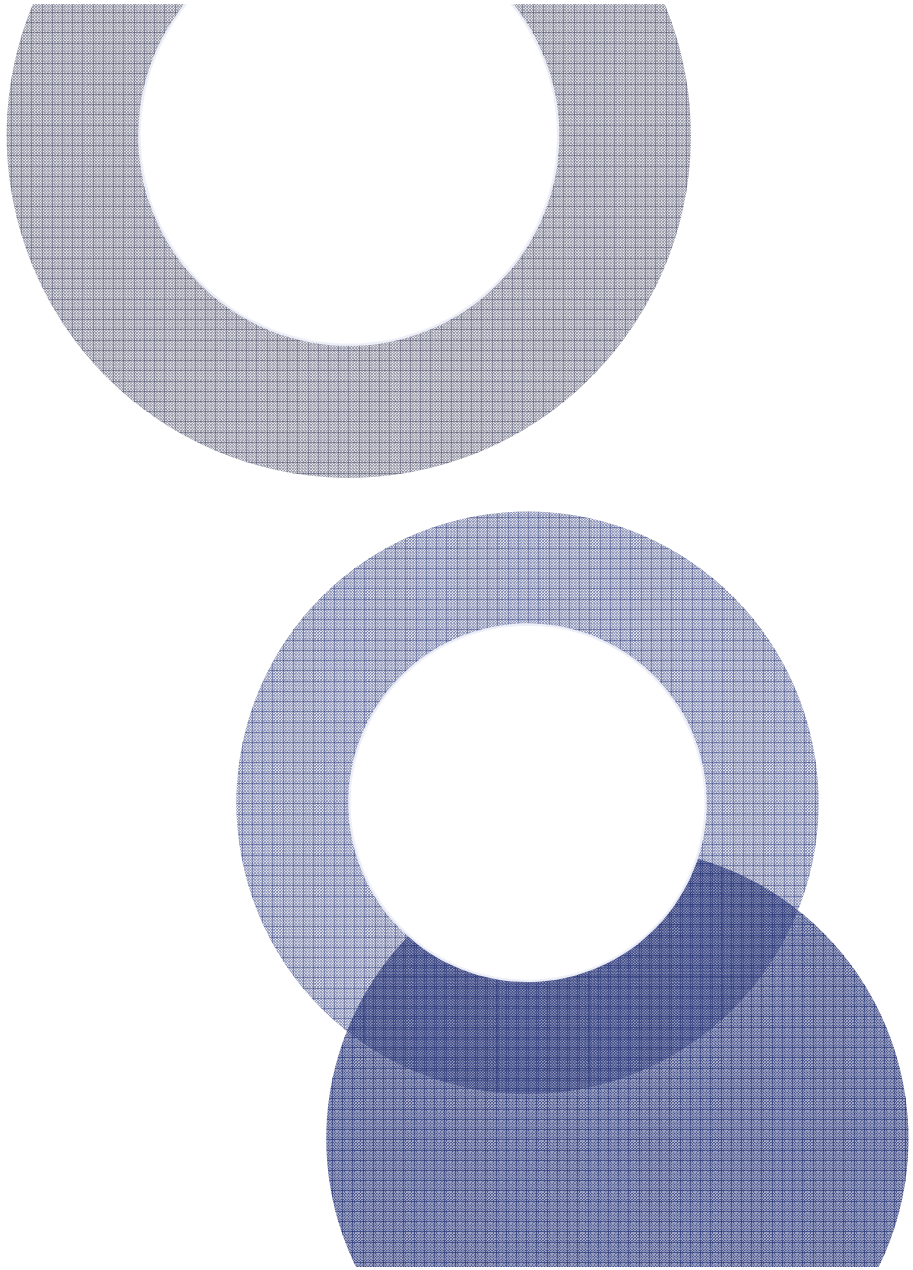
프로젝트 추진 경과

진행 예정

진행중

완료

구분	추진 내용	4월	5월	6월	7월	8월	9월	10월	11월
계획	사전 자료 조사								
분석	기존 언패커 도구 분석								
	악성 패킹 앱 분석								
개발	언패커 도구 개발								
테스트	패킹된 악성 앱 대상 테스트 진행								
마무리	논문 및 보고서 작성								
	졸업작품 발표회 및 캡스톤 공모전 준비								



II

프로젝트 개발 내용

1. 사전 연구
2. 개발 설계
3. 언패커
4. 패커 탐지

Ⅱ

프로젝트 개발 내용

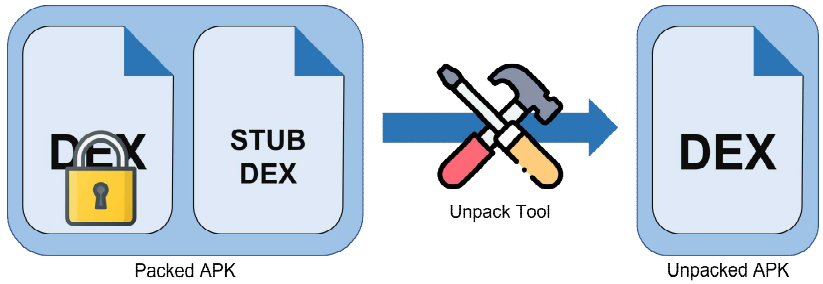
사전 연구

개발에 앞서 사전 연구 진행

패킹과 언패킹

- **패킹**이란 **원본 Dex 파일을 보호**하기 위해 사용하는 방법으로 주로 Dex 파일 은닉, Dex 파일 덤핑 방지, 안티 리버싱 등 사용
- **언패킹**이란 패킹의 반대되는 개념으로, **패킹했던 파일의 패킹을 푸는 것**

언패킹 과정



Native 언패커

- Defcon 22에서 공개된 언패커
- **프로세스**와 **스레드 메모리**에 접근 후, 시그니처를 통해 **패커의 종류를 식별**하고, Dex 파일의 Magic Number를 사용하여 **Dex 파일 추출**

Frida 언패커

- **Frida** 도구를 사용하여 제작한 언패커
- 안드로이드에서 Dex 파일을 파싱할 때 사용하는 **함수를 후킹한 후** 해당 함수로 넘어오는 **Dex 파일 추출**

15 패커 종류	원본 Dex 추출 여부	16 패커 종류	원본 Dex 추출 여부	18 패커 종류	원본 Dex 추출 여부	19 패커 종류	원본 Dex 추출 여부
15_Ali	O	16_Ali	X				
15_Baidu	X	16_Baidu	O	18_Baidu	O	19_Baidu	X
15_Bangcle	X	16_Bangcle	X	18_Bangcle	X	19_Bangcle	X
15_Ijiami	O	16_Ijiami	X	18_Ijiami	O	19_Ijiami	X
15_Qihoo	O	16_Qihoo	O	18_Qihoo	O	19_Qihoo	X
15_Tencent	X	16_Tencent	X	18_Tencent	X	19_Tencent	X

▲ Native 언패커 결과

- 언패킹이란 패킹의 반대되는 개념으로, 패킹했던 파일의 패킹을 푸는 것

안드로이드 기기에 저장한 후 분석할
앱을 실행하여 실행 파일이 이자로

15 패커 종류	원본 Dex 추출 여부	16 패커 종류	원본 Dex 추출 여부	18 패커 종류	원본 Dex 추출 여부	19 패커 종류	원본 Dex 추출 여부
15_Ali	O	16_Ali	O				
15_Baidu	O	16_Baidu	O	18_Baidu	O	19_Baidu	O
15_Bangcle	O	16_Bangcle	O	18_Bangcle	O	19_Bangcle	O
15_Ijiami	O	16_Ijiami	O	18_Ijiami	X	19_Ijiami	X
15_Qihoo	O	16_Qihoo	O	18_Qihoo	X	19_Qihoo	X
15_Tencent	O	16_Tencent	O	18_Tencent	X	19_Tencent	X

▲ Frida 언패커 결과

Ⅱ

프로젝트 개발 내용

개발 설계

✓ 사전 연구를 바탕으로 언패커 개발 설계

개발 설계

개발 목표

① 원본 **Dex File** 추출

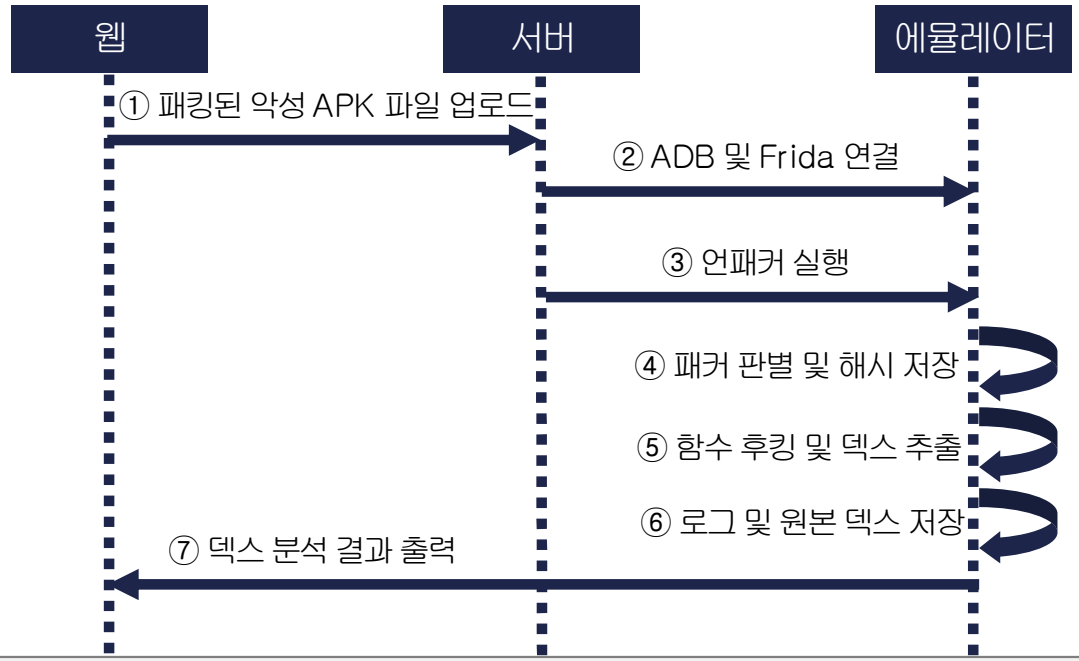
Python JavaScript

② Yara 시그니처를 응용한 **패커 식별**

③ 악성 앱 식별을 위한 MD5, SHA-1, SHA-256 **해시 값 산출**

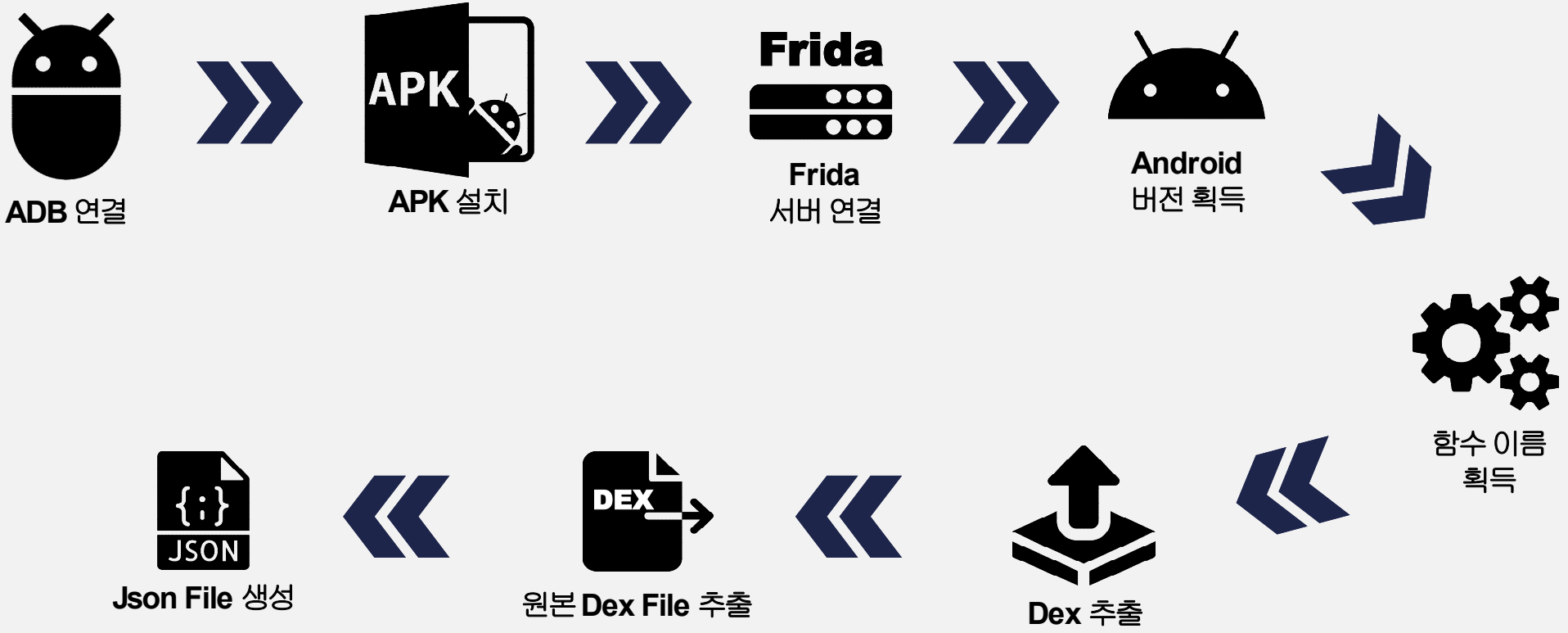
웹 사이트에 **악성 앱 파일 업로드** 후, 사용된 **패커 식별**과
탐지된 라이브러리 및 리소스 확인 후 **원본 Dex file 추출**

전체 구상도



언패커 개발

✓ 기존 언패커 분석 후 개발 진행



Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행

ADB 연결

```
def get_adb_connect(self):
    client = AdbClient()
    if not client:
        log.error_('Client connect error')

    devices = client.devices()
    if not devices:
        log.error_('Device not found')

    if len(devices) > 1:
        log.info_('Multiple devices detected, please select the device')
        for i in range(len(devices)):
            log.info_('%d : %s' %(i+1, devices[i].get_serial_no()))
        num = int(input('> '))
        self.adb = devices[num-1]
    else:
        self.adb = devices[0]
    log.info_('ADB connected : %s' %self.adb.get_serial_no())
```

ADB 연결

Pure-python-adb 패키지 이용하여 진행

```
=== Unpacking Start ===
[*] Multiple devices detected, please select the device
[*] 1 : 127.0.0.1:62026
[*] 2 : 127.0.0.1:62025
>
```

▲ 다중 디바이스 감지 시 연결 기기 선택

Json File 생성

원본 Dex File 추출

Dex 추출

Ⅱ

프로젝트 개발 내용

언패커 개발

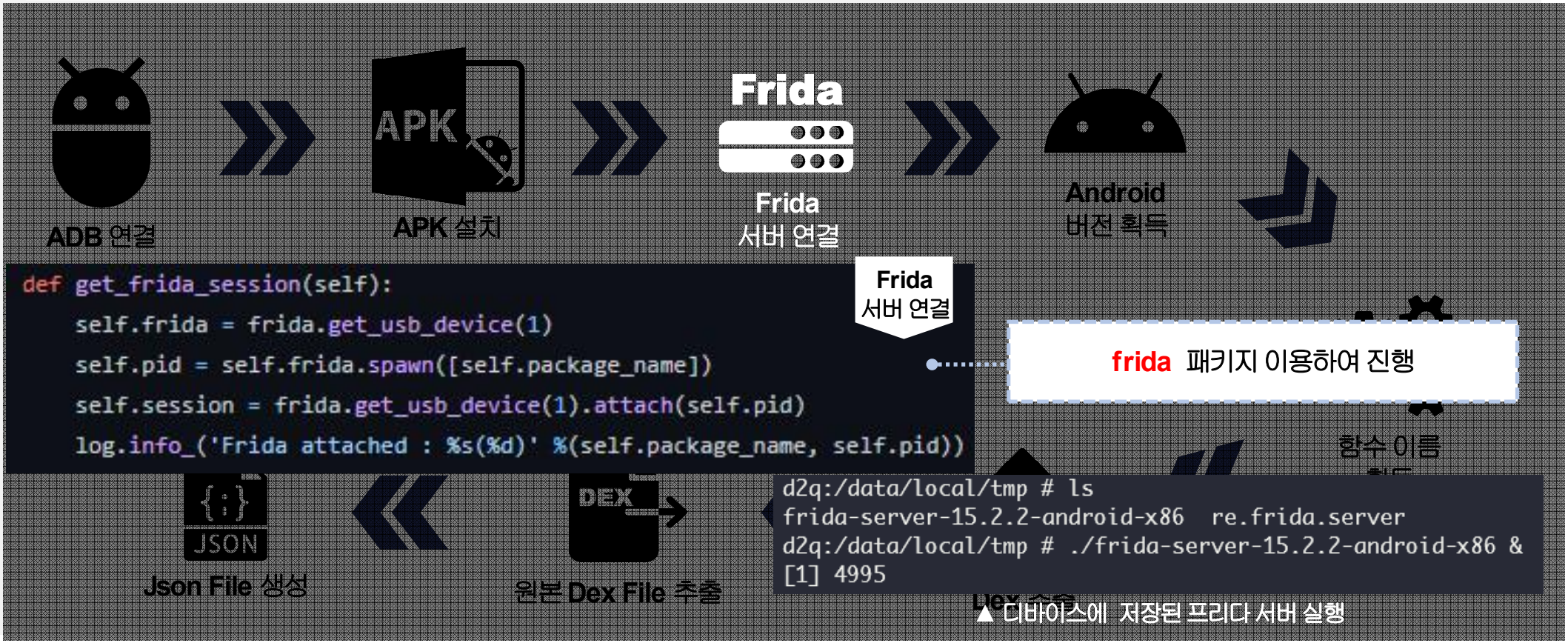
✓ 기존 언패커 분석 후 개발 진행

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행



▲ 디바이스에 저장된 프리다 서버 실행

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행


```
get_function = """
rpc.exports = {
    getFunc: function (and_ver) {
        var function_name = '';
        var i = 0;
        var android_version = parseInt(and_ver);
        if (android_version > 4) {
            var lib_name = android_version >= 10 ? 'libdexfile.so' : 'libart.so'
            var art = Module.enumerateExportsSync(lib_name);

            for(i = 0; i< art.length; i++){
                if(art[i].name.indexOf("OpenMemory") !== -1){
                    function_name = art[i].name;
                    return function_name;
                }else if(art[i].name.indexOf("OpenCommon") !== -1){
                    if (android_version >= 10 && art[i].name.indexOf("ArtDexFileLoader") !== -1)
                        continue;
                    function_name = art[i].name;
                    return function_name;
```

함수 이름 획득

후킹 시 사용할 함수 이름 탐색 및 반환

함수 이름 획득

x 추출

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행


```
get_function = """
rpc.exports = {
    getFunc: function (and_ver) {
        var function_name = '';
        var i = 0;
        var android_version = parseInt(and_ver);
        if (android_vers def get_function_name(self):
            var lib_name        script = self.session.create_script(get_function)
            var art = Mo
                                script.load()
                                api = script.exports
            for(i = 0; i
                if(art[i        self.function_name = api.get_func(self.and_ver)
                    func        log.info_('Function name : %s' %self.function_name)
                    return function_name;
                }else if(art[i].name.indexOf("OpenCommon") !== -1){
                    if (android_version >= 10 && art[i].name.indexOf("ArtDexFileLoader") !== -1)
                        continue;
                    function_name = art[i].name;
                    return function_name;
```

함수 이름 획득

rpc.exports 를 사용하여 함수 이름 저장

함수 이름 획득

x 추출

Ⅱ

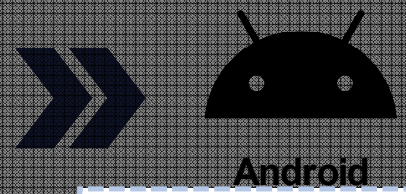
프로젝트 개발 내용

언패커 개발

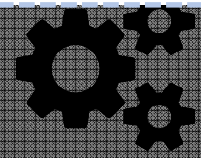
✓ 기존 언패커 분석 후 개발 진행

함수 후킹

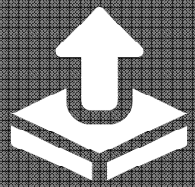
```
rpc.exports = {
    hook: function (func, proc, ver) {
        var moduleFuncName = func;
        var g_processName = proc;
        var g_AndroidOSVersion = parseInt(ver);
        console.log("[*] Dump dex start");

        if(moduleFuncName !== ""){
            var hookFunction;
            if (g_AndroidOSVersion > 4) {
                hookFunction = Module.findExportByName("libart.so", moduleFuncName);
            } else {
                hookFunction = Module.findExportByName("libdvm.so", moduleFuncName);
                if(hookFunction == null) {
                    hookFunction = Module.findExportByName("libart.so", moduleFuncName);
                }
            }
            Interceptor.attach(hookFunction,{
                onEnter: function(args){
                    var begin = 0;
                    var dexMagicMatch = false;
                    var odexMagicMatch = false;
```



함수 이름, 패키지 이름, 안드로이드 버전을
사용하여 함수 후킹 진행



함수 이름
획득



Dex 추출

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행

함수 후킹

```
Interceptor.attach(hookFunction,{
    onEnter: function(args){
        var begin = 0;
        var dexMagicMatch = false;
        var odexMagicMatch = false;

        dexMagicMatch = checkDexMagic(args[0]);
        if (dexMagicMatch === true){
            begin = args[0];
```

덱스 탐색

```
function checkDexMagic(dataAddr) {
    var magicMatch = true;
    var magicFlagHex = [0x64, 0x65, 0x78, 0x0a, 0x30, 0x33, 0x35, 0x00];

    for(var i = 0; i < 8; i++){
        if(Memory.readU8(ptr(dataAddr).add(i)) !== magicFlagHex[i]){
            magicMatch = false;
            break;
        }
    }
    return magicMatch;
}
```

함수 이름 획득

Dex 추출

후킹 후 덱스 매직 넘버 값을 통해 덱스 탐색

원본 Dex File 추출

JSON

Json File 생성

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행


```
function dumpDexToFile(isDex, begin, g_processName) {
    var dexType;
    isDex ? dexType = "dex" : dexType = "odex";
    var magic = Memory.readUtf8String(begin).replace(/\\n/g, "");
    var address = ptr(begin).add(isDex ? 0x20 : 0x1C);
    var dex_size = Memory.readInt(ptr(address));
    var dex_path = "/data/data/" + g_processName + "/" + idx + "_" + dex_size + "." + dexType;
    var dex_file = new File(dex_path, "wb");
    idx += 1;

    dex_file.write(Memory.readByteArray(begin, dex_size));
    dex_file.flush();
    dex_file.close();

    console.log("[*] magic : " + magic );
    console.log("[*] size  : " + dex_size);
    console.log("[*] dumped " + dexType + " @ " + dex_path + "\\n");
}
```

Json File 생성

Dex 추출

원본 Dex File 추출

Android
버전 획득

발견한 덱스는
"/data/data/[패키지 이름]" 경로에 저장

함수 이름
획득

Dex 추출

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행


```
def extract(self):
    out_file = self.package_name+'.tar'
    out_path = '/data/local/tmp/'+out_file
    self.adb.shell('tar cvf %s -C /data/data/%s/ .' %(out_path, self.package_name))
    self.adb.pull(out_path, out_file)
    self.adb.shell('rm '+out_path)
    self.adb.uninstall(self.package_name)
```

원본 Dex File 추출

Android 버전 획득

추출한 덱스를 포함한 "/data/data/[패키지 이름]" 경로에 모든 파일 추출

함수 이름 획득

Dex 추출

원본 Dex File 추출

Json File 생성

Ⅱ

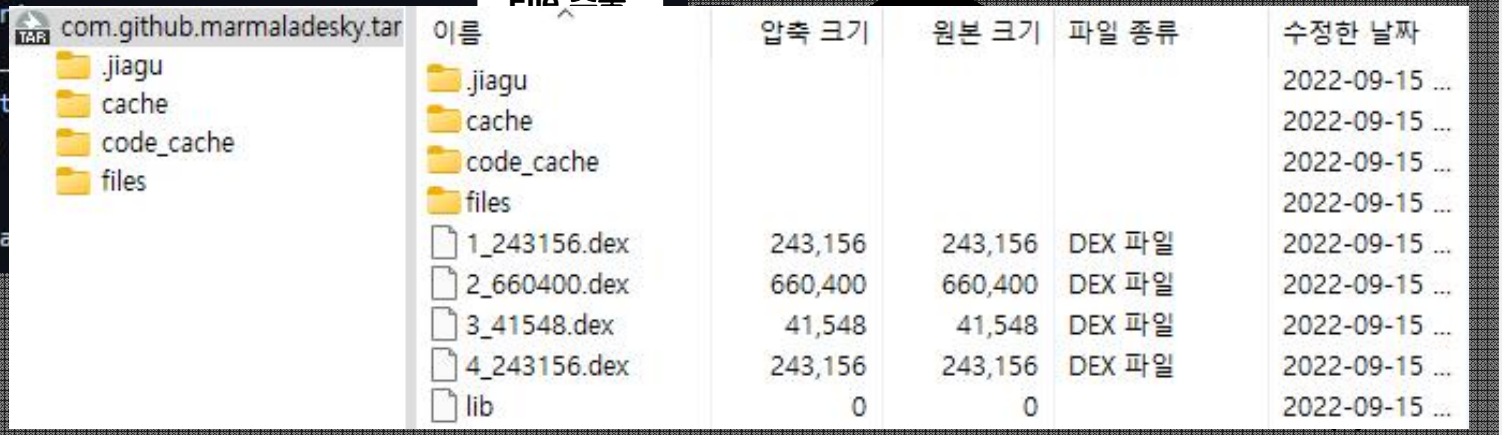
프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행


```
def extract(self):
    out_file = self.package_name+'.tar
    out_path = '/data/local/tmp/'+out_
    self.adb.shell('tar cvf %s -C /dat
    self.adb.pull(out_path, out_file)
    self.adb.shell('rm '+out_path)
    self.adb.uninstall(self.package_na
```

원본 Dex File 추출



▲ 추출 예시

함수 이름 획득

Dex 추출

원본 Dex File 추출

Json File 생성

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행


```
def json_export(self):
    self.json_log['android_version'] = self.and_ver
    self.json_log['package_name'] = self.package_name
    self.json_log['process_name'] = self.process_name
    self.json_log['function_name'] = self.function_name
```

획득한 정보 저장

ADB 연결

APK 설치

```
for dex in dex_list:
    self.json_log['dex'][dex] = {}
    obj = re.search('[0-9]_([0-9]*)\.([a-z]*)', dex)
    self.json_log['dex'][dex]['size'] = obj.group(1)
    self.json_log['dex'][dex]['type'] = obj.group(2)
    self.json_log['dex'][dex]['hash'] = {}
    self.json_log['dex'][dex]['strings'] = []
    with tr.extractfile('./'+dex) as f:
        data = f.read()
        self.json_log['dex'][dex]['hash']['md5'] = hashlib.md5(data).hexdigest()
        self.json_log['dex'][dex]['hash']['sha1'] = hashlib.sha1(data).hexdigest()
        self.json_log['dex'][dex]['hash']['sha256'] = hashlib.sha256(data).hexdigest()
        for s in re.findall(b"[\x1f-\x7e]{6,}", data):
            self.json_log['dex'][dex]['strings'].append(s.decode('utf-8'))
        for s in re.findall(b"(?:[\x1f-\x7e][\x00]){6,}", data):
            self.json_log['dex'][dex]['strings'].append(s.decode('utf-16le'))
```

문자열 및 해시 저장

JSON

Json File 생성

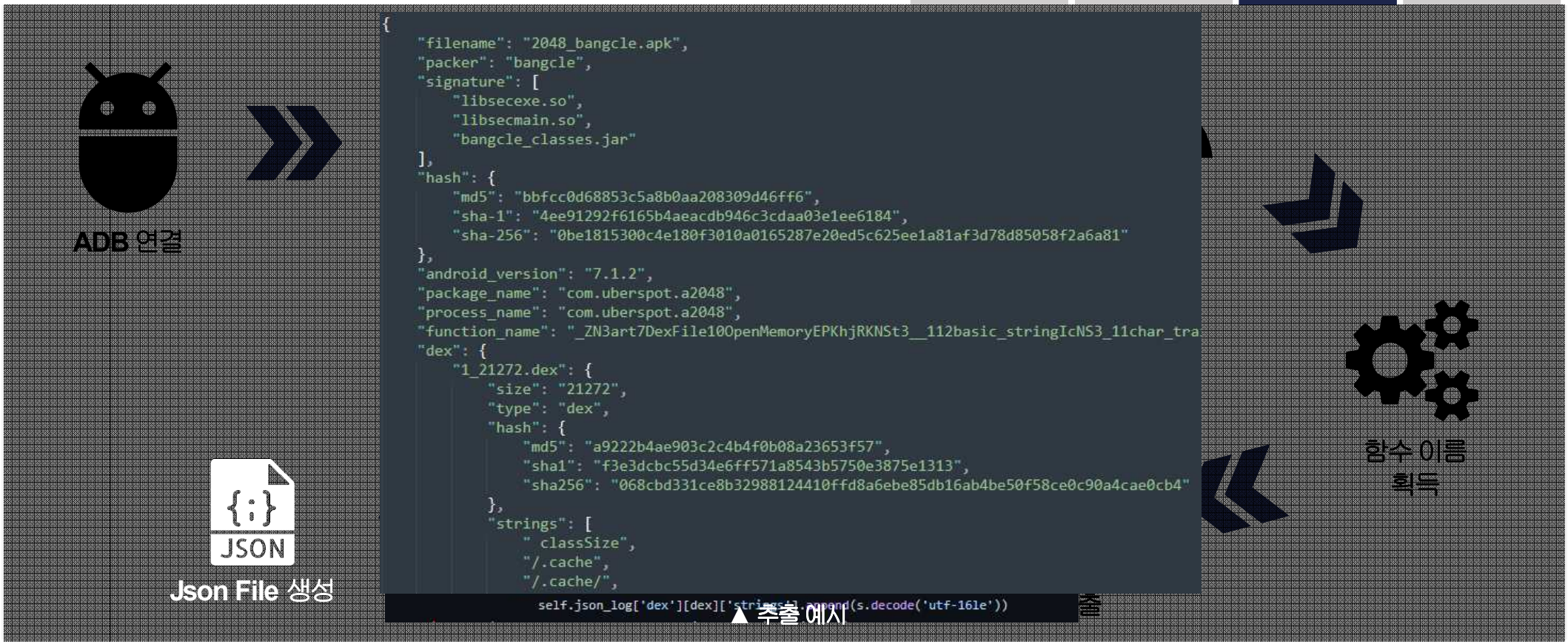
추출한 덱스 파일에서
문자열 및 해시값 추출

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행



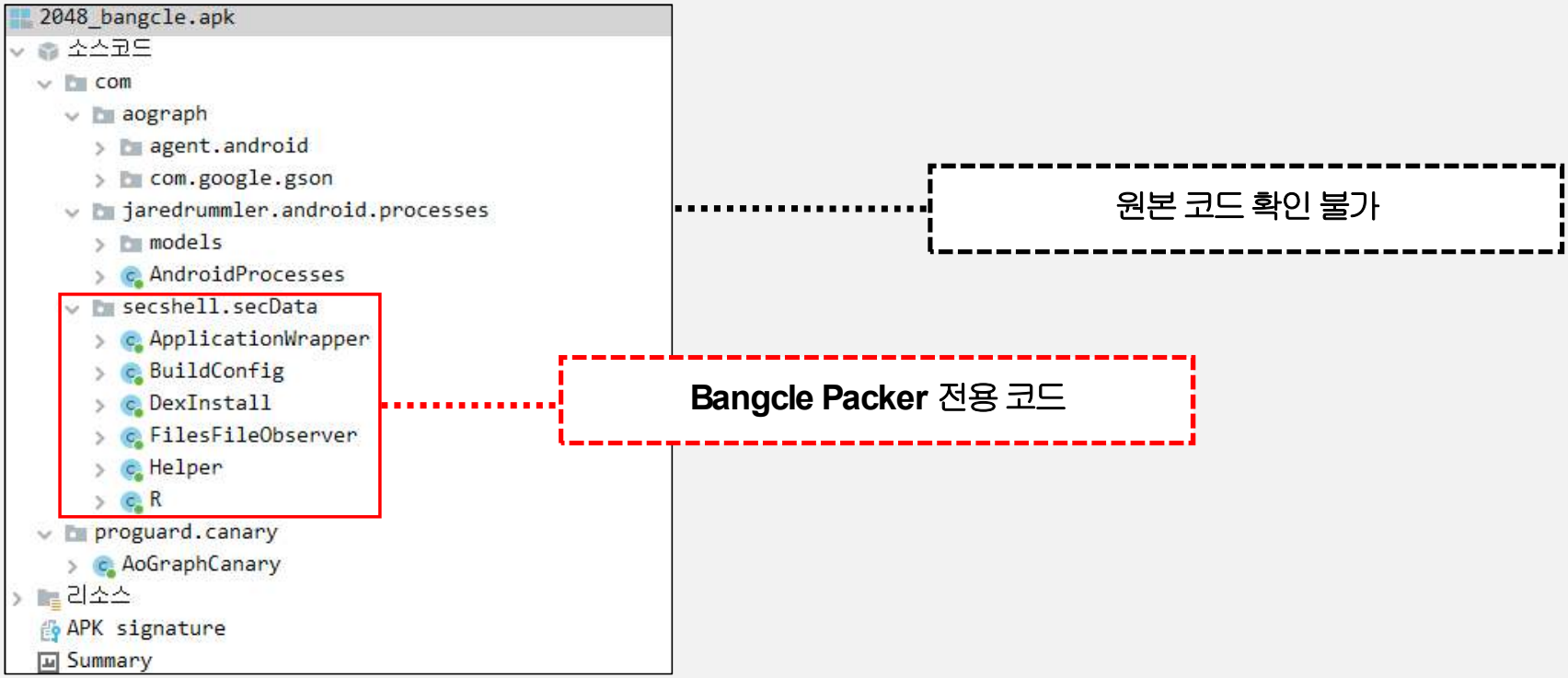
▲ 추출 예시

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행



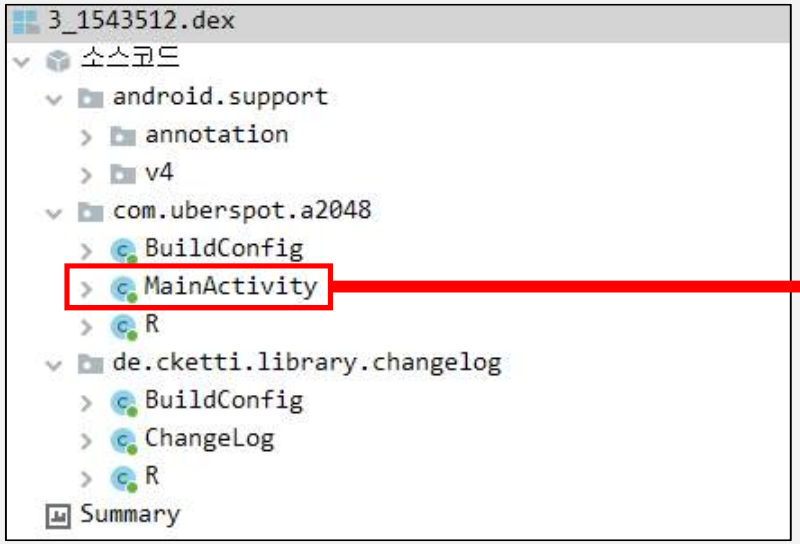
▲ 패킹된 APK

Ⅱ

프로젝트 개발 내용

언패커 개발

✓ 기존 언패커 분석 후 개발 진행



▲ 추출한 덱스

```java
public class MainActivity extends Activity {
    private static boolean DEF_FULLSCREEN = true;
    private static final String IS_FULLSCREEN_PREF = "is_fullscreen_pref";
    private static final String MAIN_ACTIVITY_TAG = "2048_MainActivity";
    private static final long mBackPressThreshold = 3500;
    private static final long mTouchThreshold = 2000;
    private long mLastBackPress;
    private long mLastTouch;
    private WebView mWebView;
    private Toast pressBackToast;

    @Override // android.app.Activity
    @SuppressLint({"SetJavaScriptEnabled", "NewApi", "ShowToast"})
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(1);
        if (Build.VERSION.SDK_INT >= 11) {
            getWindow().setFlags(ViewCompat.MEASURED_STATE_TOO_SMALL, ViewCompat.MEASURED_STATE_TOO_SMALL);
        }
        applyFullScreen(isFullScreen());
        boolean isOrientationEnabled = false;
        try {
            if (Settings.System.getInt(getContentResolver(), "accelerometer_rotation") == 1) {
                isOrientationEnabled = true;
            } else {
                isOrientationEnabled = false;
            }
        } catch (Settings.SettingNotFoundException e) {
            Log.d(MAIN_ACTIVITY_TAG, "Settings could not be loaded");
        }
        int screenLayout = getResources().getConfiguration().screenLayout & 15;
        if ((screenLayout == 3 || screenLayout == 4) && isOrientationEnabled) {
            setRequestedOrientation(4);
        }
        setContentView(R.layout.activity_main);
        ChangeLog cl = new ChangeLog(this);
```

▲ MainActivity

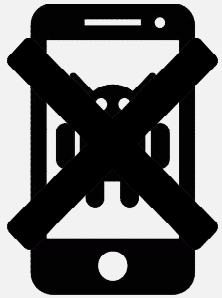
Ⅱ

프로젝트 개발 내용

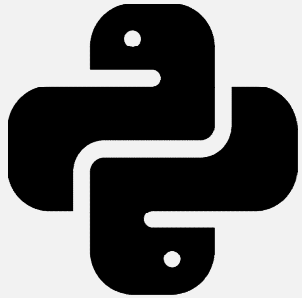
패커 탐지

✓ 패커 탐지 기능 소개

기존 개발된
Yara 도구



문제점 분석



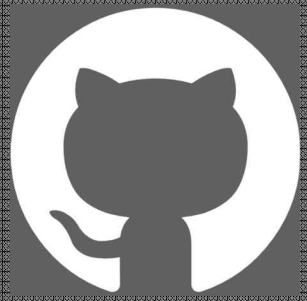
문제점 개선

Ⅱ

프로젝트 개발 내용

패커 탐지

✓ 패커 탐지 기능 소개



기존 개발된
Yara 도구

APKiD
Yara-Rules

```
rule alibaba : packer
{
  meta:
    description = "Alibaba"

  strings:
    $lib = "libmobisec.so"

  condition:
    is_apk and $lib
}
```

meta : APK에 관한 정보, 해시 등 코멘트

strings : 탐지되는 시그니처 (문자열 **or Hex)**

condition : 조건의 참/ 거짓 판별

Ⅱ

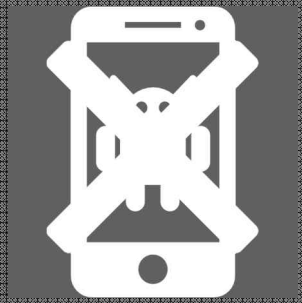
프로젝트 개발 내용

패커 탐지

✓ 패커 탐지 기능 소개

패커 중복 탐지

```
is_apk 2048_ali.apk
0x0:$zip_head: PK
0x51e40:$manifest: AndroidManifest.xml
0xb031d:$manifest: AndroidManifest.xml
tencent 2048_ali.apk
0x6b1d9:$zip_lib: lib/armeabi/libmobisecy.so
0xb0746:$zip_lib: lib/armeabi/libmobisecy.so
alibaba 2048_ali.apk
0x56eb8:$lib: libmobisec.so
0xb070b:$lib: libmobisec.so
```



문제점 분석

시그니처 중복 출력

```
is_apk 2048_ali.apk
0x0:$zip_head: PK
0x51e40:$manifest: AndroidManifest.xml
0xb031d:$manifest: AndroidManifest.xml
tencent 2048_ali.apk
0x6b1d9:$zip_lib: lib/armeabi/libmobisecy.so
0xb0746:$zip_lib: lib/armeabi/libmobisecy.so
alibaba 2048_ali.apk
0x56eb8:$lib: libmobisec.so
0xb070b:$lib: libmobisec.so
```

일부 패커는 중복 탐지로 판별 불가

탐지된 시그니처 중복 제거 출력 옵션 없음

Ⅱ

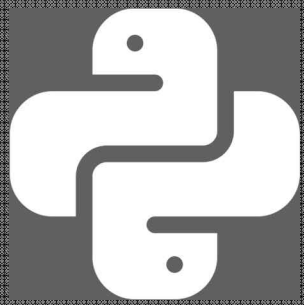
프로젝트 개발 내용

패커 탐지

✓ 패커 탐지 기능 소개

패커 탐지

```
# alibaba packer
alibaba = yara.compile(
    source='rule alibaba \
    {strings:$lib = "libmobisec.so" \
    condition: $lib}'
)
alibaba_match = alibaba.match(filename)
```



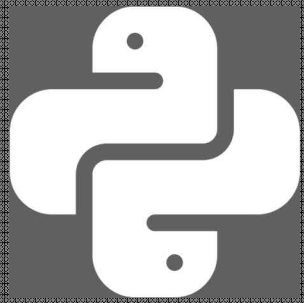
문제점 개선

일부 패커 중복 탐지와 시그니처 중복 제거 등
문제점 개선과 편의성 제공 **Yara** 개발

✓ 패커 탐지 기능 소개

시그니처 중복 제거

```
if len(apk_match) == 1 and len(alibaba_match) == 1:
    print(f"{filename} 패커는 alibaba 패커입니다.\n")
    print(f"{filename}의 탐지된 시그니처")
    signature = alibaba_match[0].strings
    signature_list = []
    for x, y, z in signature:
        if z not in signature_list:
            signature_list.append(z)
    for i in signature_list:
        result = i.decode(encoding="utf-8")
        print(result)
```



문제점 개선

탐지되는 문자열 혹은 **Hex**의 시그니처 중복 제거

Ⅱ

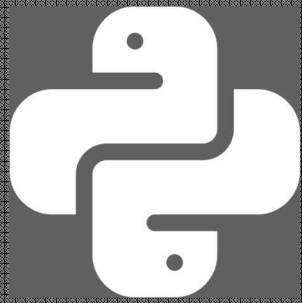
프로젝트 개발 내용

패커 탐지

✓ 패커 탐지 기능 소개

해시값 출력

```
# apk 파일 해시 출력
if len(apk_match) == 1:
    apk_file = open(filename, "rb")
    apk_data = apk_file.read()
    apk_file.close()
    print(f"{filename} 파일의 해시 값 출력 (MD5, SHA-1, SHA-256)")
    print("MD5: " + hashlib.md5(apk_data).hexdigest())
    print("SHA-1: " + hashlib.sha1(apk_data).hexdigest())
    print("SHA-256: " + hashlib.sha256(apk_data).hexdigest())
```



문제점 개선

APK 파일의 **MD5 / SHA-1 / SHA-256** 해시값 출력

Ⅱ

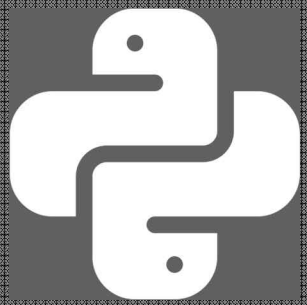
프로젝트 개발 내용

패커 탐지

✓ 패커 탐지 기능 소개

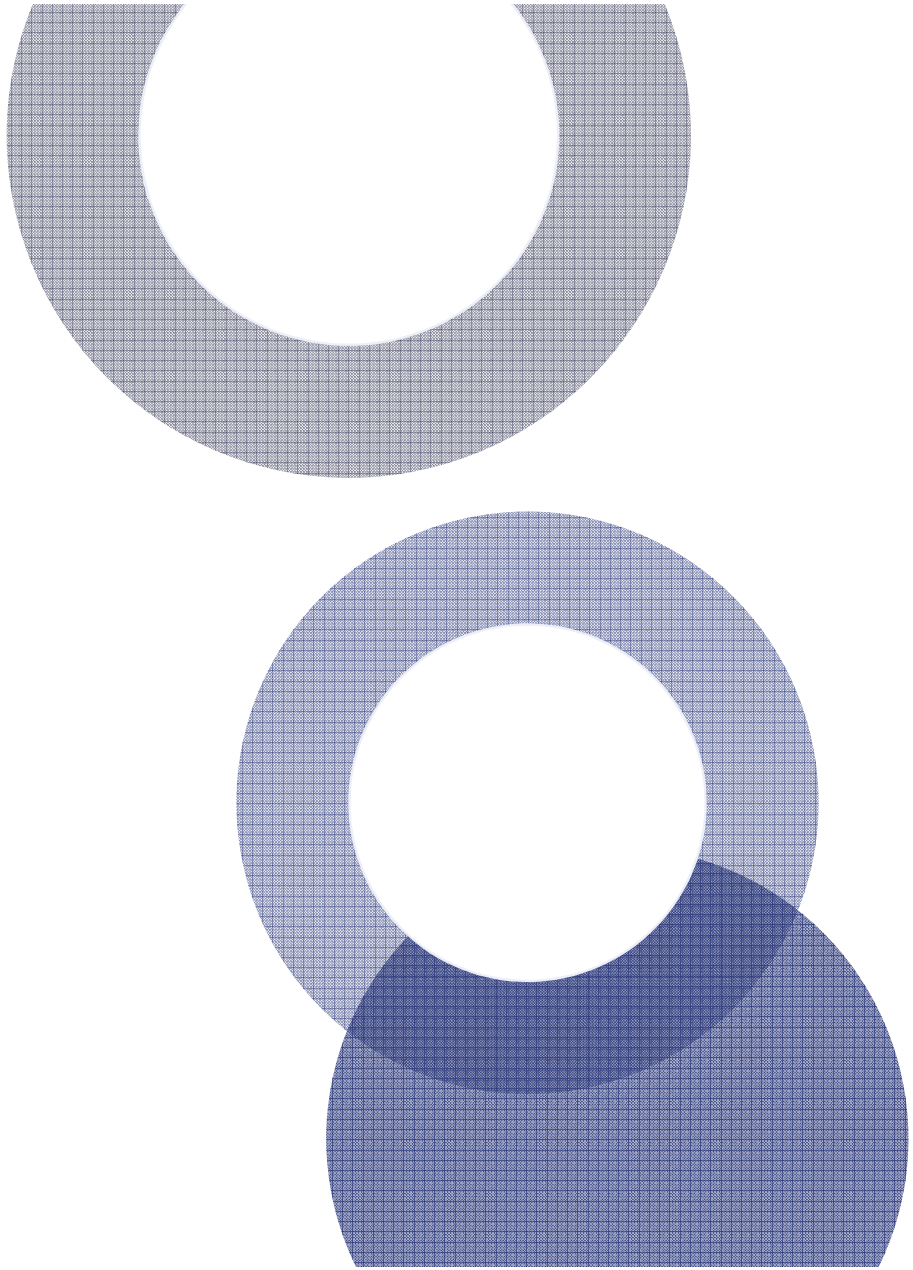
최종 출력 결과

```
D:\Yara-Unpacker>python Yara-Rules_APK.py
APK 파일명을 입력해주세요.
2048_ali.apk
정상적인 apk 파일입니다.

2048_ali.apk 패커는 alibaba 패커입니다.

2048_ali.apk의 탐지된 시그니처
libmobisec.so

2048_ali.apk 파일의 해시 값 출력 (MD5, SHA-1, SHA-256)
MD5: 652bda9f7b8155f413577520e0bda6fc
SHA-1: 4ccd3909dba290789d338035b8400d1d4dbe8e9a
SHA-256: 9e529c4e3402b208f955b1fb6fddd493816fccc48ac1981fd0a1aed35d85acb3
```



문제점 개선

APK 판별 및 탐지된 시그니처 기반 **패커 판별**
악성 앱 분석을 위한 **해시 값 산출**



III

프로젝트 결론

1 프로젝트 결과

Ⅲ

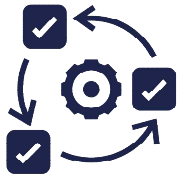
프로젝트 결론

결과

✓ 프로젝트 산출물 및 최종 결론

패킹된 악성 앱 분석을 위한 언패커 개발

- Yara Rule과 Frida Script를 활용하여 Python 언어로 개발된 안드로이드 전용 언패커



악성 앱 분석 내용을 기반으로 한 분석 보고서

- 프로젝트를 진행하며 관련된 모든 내용을 노션에 정리
정리한 내용을 바탕으로 보고서 제작

연구 내용을 기반으로 한 논문 (학회 투고 및 컨퍼런스 발표)

- 패킹된 악성 앱을 언패킹하기 위한 연구 내용을 바탕으로
논문 작성 및 컨퍼런스 발표

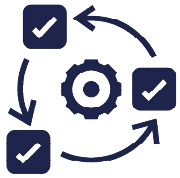
Ⅲ

프로젝트 결론

결과

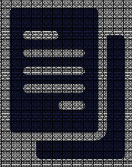
✓ 프로젝트 산출물 및 최종 결론

패킹된 악성 앱 분석을 위한 언패커 개발



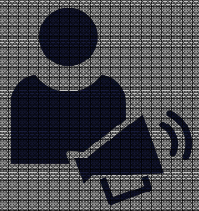
- Yara Rule과 Frida Script를 활용하여 Python 언어로 개발된 안드로이드 전용 언패커

악성 앱 분석 내용을 기반으로 한 분석 보고서



- 프로젝트를 진행하며 관련된 모든 내용을 노션에 정리
정리한 내용을 바탕으로 보고서 제작

연구 내용을 기반으로 한 논문 (학회 투고 및 컨퍼런스 발표)



- 패킹된 악성 앱을 언패킹하기 위한 연구 내용을 바탕으로 논문 작성 및 컨퍼런스 발표

Ⅲ 프로젝트 결론

결과

✓ 프로젝트 산출물 및 최종 결론


```
C:\Users\okko2\OneDrive\Desktop\git\Android-Unpacker-Project\unpacker>python main.py 2048_bangcle.apk
정상적인 apk 파일입니다.

2048_bangcle.apk 패커는 bangcle 패커입니다.

=== 2048_bangcle.apk의 탐지된 시그니처 ===
libsecexe.so
libsecmain.so
bangcle_classes.jar

2048_bangcle.apk 파일의 해시 값 출력 (MD5, SHA-1, SHA-256)
MD5: bbfcc0d68853c5a8b0aa208309d46ff6
SHA-1: 4ee91292f6165b4aeacdb946c3cdaa03e1ee6184
SHA-256: 0be1815300c4e180f3010a0165287e20ed5c625ee1a81af3d78d85058f2a6a81

=== Unpacking Start ===
[*] Multiple devices detected, please select the device
[*] 1 : 127.0.0.1:62026
[*] 2 : 127.0.0.1:62025
> 2
[*] ADB connected : 127.0.0.1:62025
[*] 2048_bangcle.apk install success
[*] Frida attached : com.uberspot.a2048(5436)
[*] Android version : 7.1.2
[*] Function name : _ZN3art7DexFile10OpenMemoryEPKhjRKNSt3__112basic_stringIcNS3_11char_traitsIcEENS3
EEEjPNS_6MemMapEPKNS_10OatDexFileEPS9_
[*] Process name : com.uberspot.a2048
[*] Dump dex start
[*] magic : dex035
[*] size  : 21272
[*] dumped dex @ /data/data/com.uberspot.a2048/1_21272.dex
```

▲ 언패커 툴 출력 화면

com.github.marmaladesky.tar

- .jiagu
- cache
- code_cache
- files

이름	압축 크기	원본 크기	파일 종류
.jiagu			
cache			
code_cache			
files			
1_243156.dex	243,156	243,156	DEX 파일
2_660400.dex	660,400	660,400	DEX 파일
3_41548.dex	41,548	41,548	DEX 파일
4_243156.dex	243,156	243,156	DEX 파일
lib	0	0	

▲ 추출한 원본 덱스 및 관련 파일(tar)


```
{
    "filename": "2048_bangcle.apk",
    "packer": "bangcle",
    "signature": [
        "libsecexe.so",
        "libsecmain.so",
        "bangcle_classes.jar"
    ],
    "hash": {
        "md5": "bbfcc0d68853c5a8b0aa208309d46ff6",
        "sha-1": "4ee91292f6165b4aeacdb946c3cdaa03e1ee6184",
        "sha-256": "0be1815300c4e180f3010a0165287e20ed5c625ee1a81af3d78d85058f2a6a81"
    },
```


▲ 언패킹 결과(Json)

III

프로젝트 결론

결과

✓ 프로젝트 산출물 및 최종 결론

패킹된 악성 앱 분석을 위한 언패

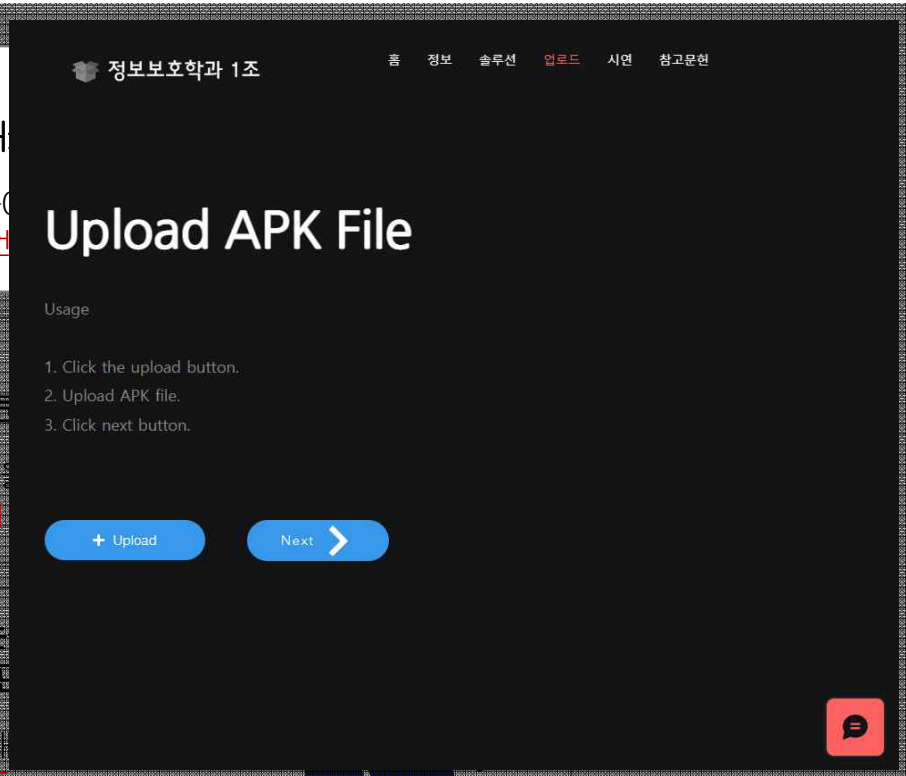
-Yara Rule과 Frida Script를 활용하
언어로 개발된 안드로이드 전용 언

악성 앱 분석 내용을 기반으로 한

- 프로젝트를 진행하며 관련된 모든 내용
정리한 내용을 바탕으로 보고서

연구 내용을 기반으로 한 논
(학회 투고 및 컨퍼런스 발

- 패킹된 악성 앱을 언패킹하기 위한 연구 내
논문 작성 및 컨퍼런스 발표

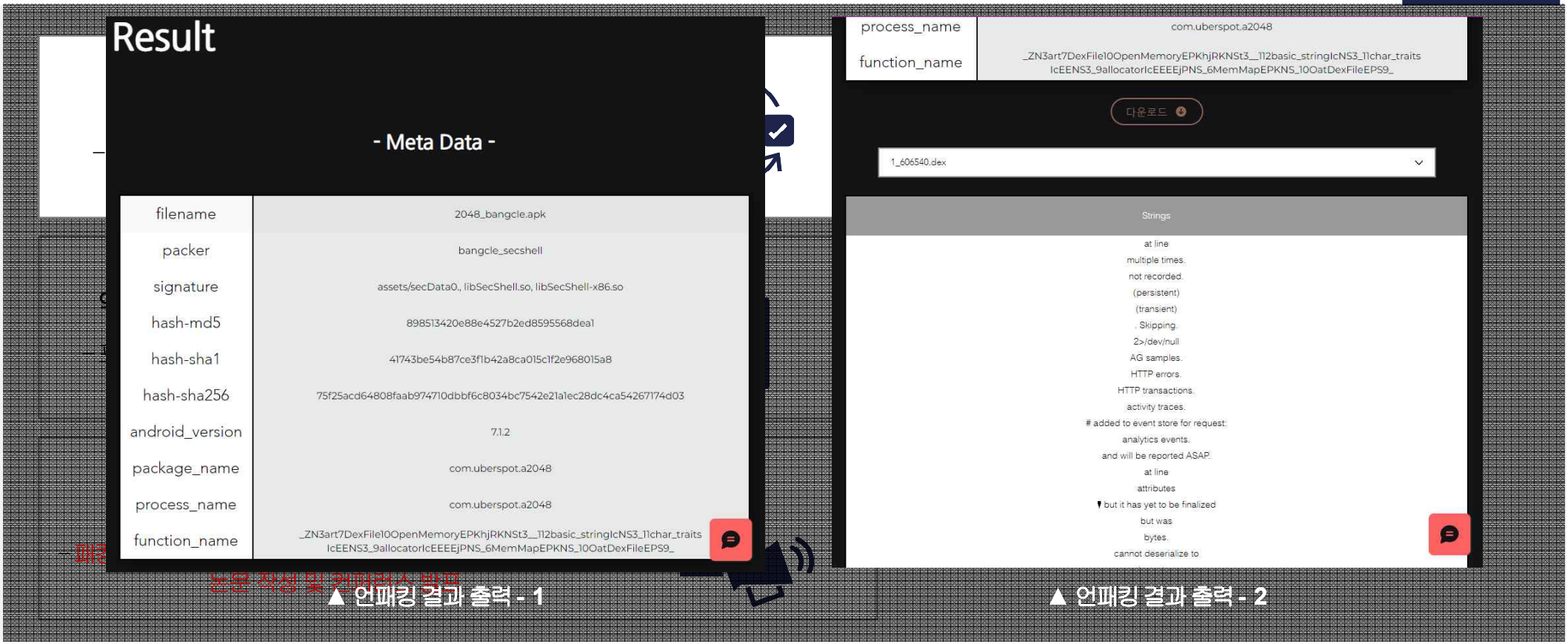


▲ APK 파일 업로드

Ⅲ 프로젝트 결론

결과

✓ 프로젝트 산출물 및 최종 결론



▲ 언패킹 결과 출력 - 1

▲ 언패킹 결과 출력 - 2

Ⅲ

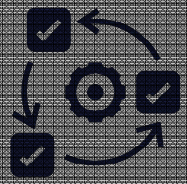
프로젝트 결론

결과

✓ 프로젝트 산출물 및 최종 결론

패킹된 악성 앱 분석을 위한 언패커 개발

- Yara Rule과 Frida Script를 활용하여 Python 언어로 개발된 안드로이드 전용 언패커

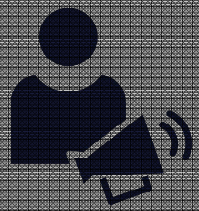


악성 앱 분석 내용을 기반으로 한 분석 보고서

- 프로젝트를 진행하며 관련된 모든 내용을 노션에 정리
정리한 내용을 바탕으로 보고서 제작

연구 내용을 기반으로 한 논문
(학회 투고 및 컨퍼런스 발표)

- 패킹된 악성 앱을 언패킹하기 위한 연구 내용을 바탕으로
논문 작성 및 컨퍼런스 발표



Ⅲ 프로젝트 결론

결과

✓ 프로젝트 산출물 및 최종 결론

주차별 자료정리

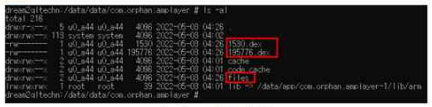
Table + Add view

Name	Tags	Property
배경지식, 패커관련	동훈	April 11, 2022
배경지식	민영	April 11, 2022
배경지식 및 자료조사	유민	April 11, 2022
언패커 사용	동훈	April 19, 2022
DexHunter	민영	April 19, 2022
AppSpear 논문 번역 및 정리	유민	April 19, 2022
android unpacker 분석	동훈	May 3, 2022
Frida 언패킹 테..스트	민영	May 3, 2022
Frida 언패킹 테스트	유민	May 3, 2022
언패커 자동화 툴 개발	동훈	May 10, 2022
dexdump.js 코드 분석 및 언패킹 테스트 분석	민영	May 10, 2022
Frida 언패킹 분석 및 정리	유민	May 10, 2022
native unpacker 분석 결과	동훈	May 17, 2022
Frida 언패킹 오류 분석	민영	May 17, 2022
Frida 언패킹 Dex 분석	유민	May 17, 2022

+ New

▲ 주차별 정리 목록

언패킹 결과 확인 (Dex 확인)



[그림 46 언패킹 결과]

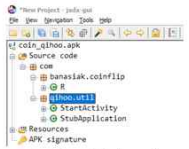
adb pull → Dex 파일 로컬로 가져오기

adb pull 명령어를 이용하여 Dex 파일을 로컬 컴퓨터로 복사하여 가져온다.

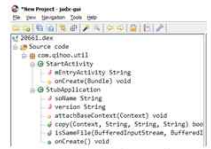
[그림 47 언패킹 결과 가져온 후]

2.9.3 사용 결과

Frida 언패킹을 통하여 원본 Dex가 추출되지 않는 예시이다. 언패킹이 제대로 이루어지지 않아 Stub 함수를 가진 Dex나 암호화 된 Dex가 그대로 추출된다. 다음 [그림 48], [그림 49]는 15 버전의 qihoo 패커가 사용된 coin.apk를 예시로 하였다. 확인해보면 해당 Dex는 원본 Dex가 아니라는 것을 예시를 통하여 확인할 수 있다.



[그림 48 패킹된 앱 모습]

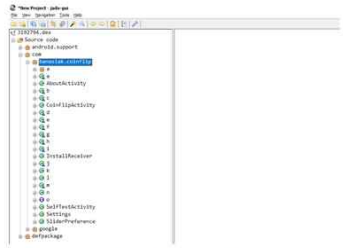


[그림 49 언패킹 후 모습]

이와 반대로 다음 패커는 정상적으로 원본 Dex가 추출된다.

확인해보면 apk는 암호화 된 Dex와 함께 Stub 함수가 존재하게 되며, 정상적으로 언패킹 된 Dex를 확인해보면 Stub 함수가 존재하지 않고 해당 패키지의 원본 코드로 보아는 것을 예시를 통하여 확인할 수 있다.

- 29 -



[그림 50 정상적으로 언패킹된 모습]

2.9.4 오류 분석

Process crashed: Trace/BPT trap – Android 7

- frida가 돌고 있는 게 탐지되면 종료시키는 에러

Process crashed: java.lang.UnsatisfiedLinkError – Android 7

- 버전 문제일 가능성이 큼
- libbaiduprotect_x86.so
- libsecexe.x86.so → 앱을 보호하는 Package 파일의 역할이 주로 이 파일을 사용함. (libsecexe.x86.so에 보호 모듈 존재)
- libexec.so → 버전 문제
- libprotectClass.so has unexpected e_machine: 40
- libbaiduprotect_x86.so → 얘가 대상 앱의 dex를 강화하고 패킹함 그래서 원본 엑스를 추출할 수 없음
- "/data/data/com.uberspot.a2048/.lib/libexec.so" has invalid shdr offset/size: 1177/21000 → 패킹할 때 헤더가 변경되지 않았을 때나 -android-shlib를 추가하지 않았을 때 발생하는 오류
- cannot locate symbol "__stack_chk_guard" referenced by "/data/data/앱패키지명/files/libjiagu.so"

Process crashed: java.lang.ClassNotFoundException – Android 7

- Didn't find class "com.banasiak.coinflip.CoinFlip" on path: DexPathList[[zip file "/data/app/com.banasiak.coinflip-1/base.apk"],nativeLibraryDirectories=[/data/app/com

- 30 -

▲ 최종 자료 정리

Ⅲ 프로젝트 결론
결과

✓ 프로젝트 산출물 및 최종 결론

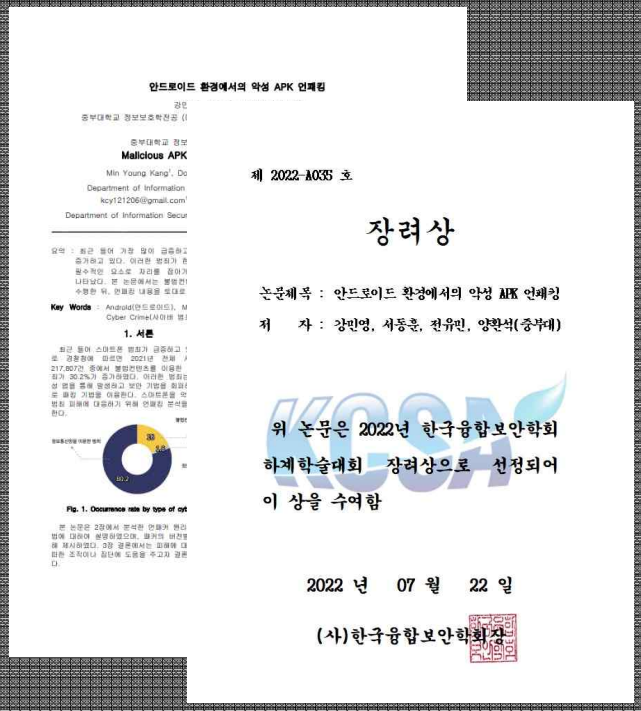
패킹된 악성 앱 분석을 위한 언패커 개발

▲ 한이음ICT 멘토링 진행 및 논문 투고

- 프로젝트를 진행하며 관련된 모든 내용을 노션에 정리
정리한 내용을 바탕으로 보고서 제작

연구 내용을 기반으로 한 논문
(학회 투고 및 컨퍼런스 발표)

- 패킹된 악성 앱을 언패킹하기 위한 연구 내용을 바탕으로
논문 작성 및 컨퍼런스 발표



▲ 학회 논문 장려상 수여

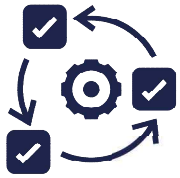
Ⅲ 프로젝트 결론

결과

✓ 프로젝트 산출물 및 최종 결론

패킹된 악성 앱 분석을 위한 언패커 개발

- Yara Rule과 Frida Script를 활용하여 Python 언어로 개발된 안드로이드 전용 언패커



악성 앱 분석 내용을 기반으로 한 분석 보고서

- 프로젝트를 진행하며 관련된 모든 내용을 노션에 정리
정리한 내용을 바탕으로 보고서 제작

연구 내용을 기반으로 한 논문
(학회 투고 및 컨퍼런스 발표)

- 패킹된 악성 앱을 언패킹하기 위한 연구 내용을 바탕으로
논문 작성 및 컨퍼런스 발표

“ **패킹 기법**에 관한 **연구 증진** ”

감사합니다.