

# Smali Code



## INDEX

## 목차

**Chapter 1** Smali Code란?

**Chapter 2** Java to smali

**Chapter 3** Uncrackable 1

**Chapter 4** Q & A



## 01. Smali Code란?

- 안드로이드 가상머신의 바이트코드를 **사람이 읽기 쉬운** 형태로 표현한 언어
- **바이트코드 수준에서 동작**하므로 Java보다 앱의 동작을 더 세밀하게 제어할 수 있다.
- 앱 리버싱, 앱 변조...등에 사용됨

## 01. Smali Code란?

- 읽기 쉬운 코드로 나타냄

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	64	65	78	0A	30	33	35	00	50	A5	69	FB	7B	2E	76	C2	dex.035:PWt@{.vA
00000010	DD	3F	B3	76	88	3D	AF	25	D1	B8	8B	24	E4	A1	59	06	Y?'v'='N,<@a;Y.
00000020	1C	18	21	00	70	00	00	00	78	56	34	12	00	00	00	00	...!.p...xV4.....
00000030	00	00	00	00	40	17	21	00	0C	53	00	00	70	00	00	00	....@...!.S..p...
00000040	67	08	00	00	A0	4C	01	00	46	0D	00	00	3C	6E	01	00	q... L..F...<n..
00000050	98	2C	00	00	84	0D	02	00	B0	3E	00	00	44	72	03	00	~,.....°>..Dr..
00000060	70	05	00	00	C4	67	05	00	58	02	1B	00	C4	15	06	00	p...Ag...X...A...
00000070	22	08	16	00	24	08	16	00	57	08	16	00	96	08	16	00	"...\$.W...-...
00000080	99	08	16	00	A7	08	16	00	B5	08	16	00	BC	08	16	00	"...\$.p...4...
00000090	D9	08	16	00	F0	08	16	00	0E	09	16	00	16	09	16	00	Ü...8.....
000000A0	33	09	16	00	4F	09	16	00	5F	09	16	00	6A	09	16	00	3...O..._...j...
000000B0	7D	09	16	00	8D	09	16	00	A0	09	16	00	AF	09	16	00	}....._...
000000C0	CB	09	16	00	D7	09	16	00	E8	09	16	00	F9	09	16	00	È...*...è...ù...
000000D0	0A	0A	16	00	21	0A	16	00	38	0A	16	00	45	0A	16	00	....!...8...E...
000000E0	50	0A	16	00	68	0A	16	00	7F	0A	16	00	93	0A	16	00	P...h....."
000000F0	A9	0A	16	00	BC	0A	16	00	D0	0A	16	00	DE	0A	16	00	@...4...8...P...
00000100	E1	0A	16	00	E5	0A	16	00	EA	0A	16	00	F0	0A	16	00	â...â...è...8...
00000110	F5	0A	16	00	0C	0B	16	00	1C	0B	16	00	36	0B	16	00	ô.....6...

class.dex



```
# direct methods
.method public constructor <init>(Landroid/content/Context;)V
    .locals 1

    const-string v0, "context"

    invoke-static {p1, v0}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullParameter(Ljava/lang/Object;

    .line 14
    invoke-direct {p0}, Landroidx/appcompat/app/AppCompatActivity;-><init>()V

    .line 17
    iput-object p1, p0, Lcom/playground/anditer/DebuggerDetector;->context:Landroid/content/Context;

    return-void
.end method
```

.smali

## 01. Smali Code란?

- APK Easy Tool, baksmali를 통해 얻을 수 있음



이름	수정한 날짜	유형	크기
build	2024-11-14 오후 5:27	파일 폴더	
original	2024-11-13 오후 10:20	파일 폴더	
res	2024-11-13 오후 10:20	파일 폴더	
smali	2024-11-13 오후 10:20	파일 폴더	
AndroidManifest.xml	2024-11-13 오후 10:20	XML 문서	1KB
apktool.yml	2024-11-13 오후 10:20	YML 파일	1KB

## 01. Smali Code란?

- 코드패치 과정



## 02. Java to Smali

```
public static int addInteger2(int num1, int num2){  
    int res;  
    res = num1 + num2;  
    return res;  
}
```

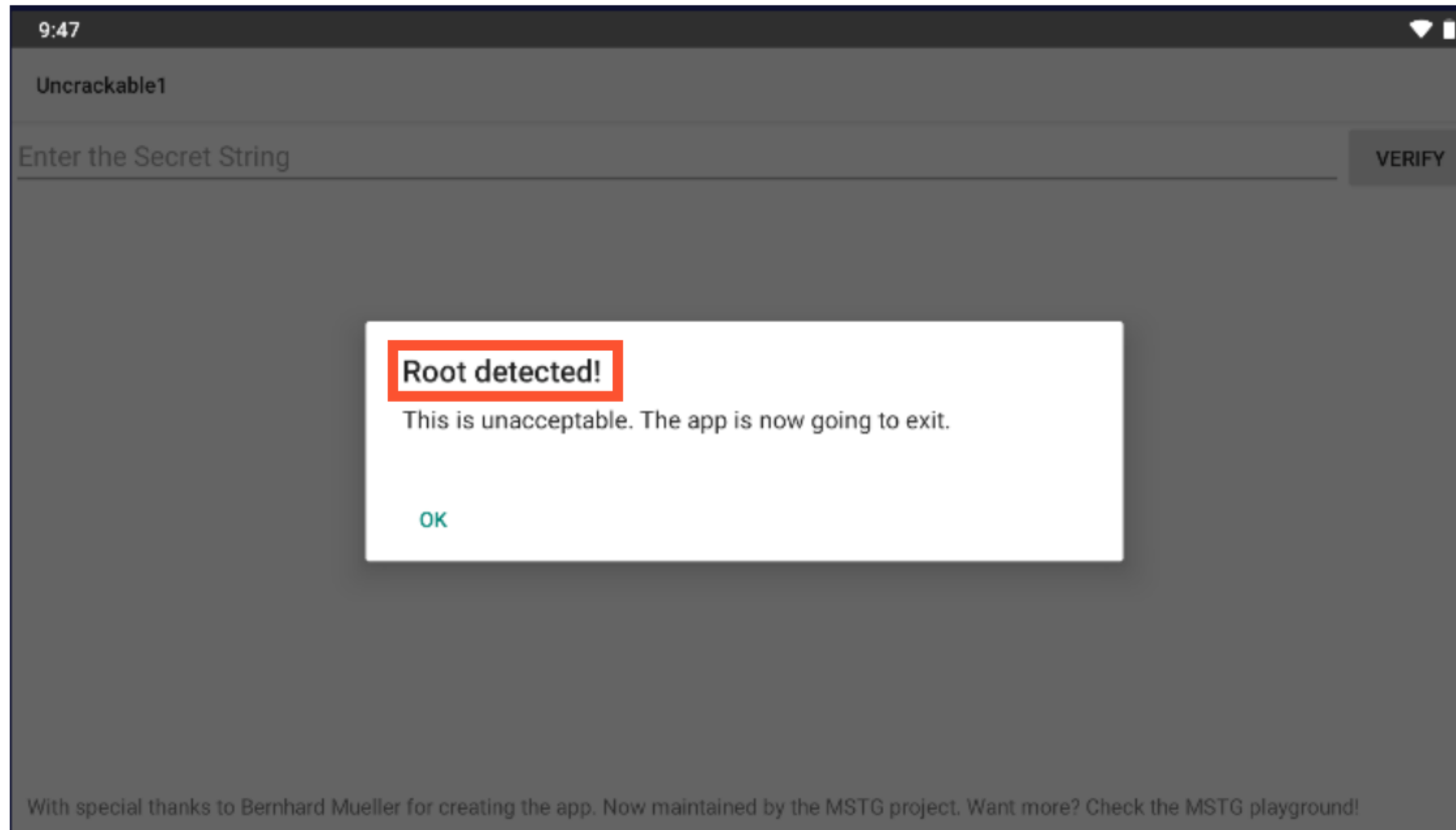
Java

```
.method public static addInteger2(II)I  
    .locals 1  
    .param p0, "num1"    # I  
    .param p1, "num2"    # I  
  
    .line 13  
    add-int v0, p0, p1  
  
    .line 14  
    .local v0, "res":I  
    return v0  
.end method
```

Smali

## 03. 루팅 탐지 우회

- 앱 실행 즉시 루팅 탐지





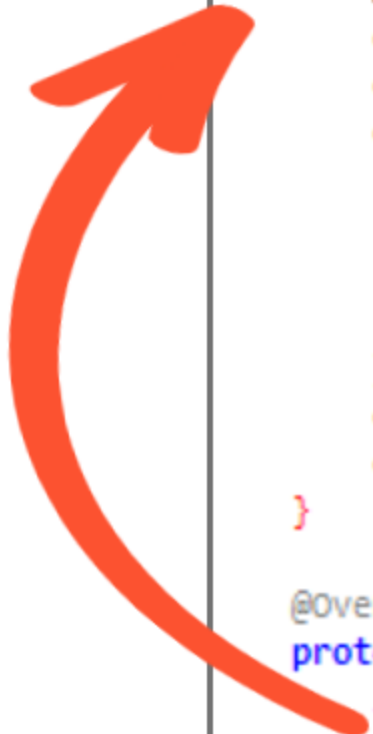
## 03. 루팅 탐지 우회

- Jadx로 먼저 디컴파일 후 소스코드 확인

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    package="owasp.mstg.uncrackable1">
    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="28"/>
    <application
        android:theme="@style/AppTheme"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:allowBackup="true">
        <activity
            android:label="@string/app_name"
            android:name="sg.vantagepoint.uncrackable1.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

## 03. 루팅 탐지 우회

- 가장 먼저 실행되는 onCreate 함수



```
public class MainActivity extends Activity {}  
    private void a(String str) {  
        AlertDialog create = new AlertDialog.Builder(this).create();  
        create.setTitle(str);  
        create.setMessage("This is unacceptable. The app is now going to exit.");  
        create.setButton(-3, "OK", new DialogInterface.OnClickListener() { // from class: sg.vantagepoint.uncrackable1.MainActivity.1  
            @Override // android.content.DialogInterface.OnClickListener  
            public void onClick(DialogInterface dialogInterface, int i) {  
                System.exit(0);  
            }  
        });  
        create.setCancelable(false);  
        create.show();  
    }  
  
    @Override // android.app.Activity  
    protected void onCreate(Bundle bundle) {  
        if (c.a() || c.b() || c.c()) {  
            a("Root detected!");  
        }  
        if (b.a(getApplicationContext())) {  
            a("App is debuggable!");  
        }  
        super.onCreate(bundle);  
        setContentView(R.layout.activity_main);  
    }
```

## 03. 루팅 탐지 우회

- 가장 먼저 실행되는 onCreate 함수

```
public class c {  
    public static boolean a() {  
        for (String str : System.getenv("PATH").split(":")) {  
            if (new File(str, "su").exists()) {  
                return true;  
            }  
        }  
        return false;  
    }  
  
    public static boolean b() {  
        String str = Build.TAGS;  
        return str != null && str.contains("test-keys");  
    }  
  
    public static boolean c() {  
        for (String str : new String[]{"/system/app/Superuser.apk", "/system/xbin/daemonsu", "/system/etc/init.d/99SuperSUDaemon", "/system/bin/.ext/.su"})  
            if (new File(str).exists()) {  
                return true;  
            }  
        }  
        return false;  
    }  
}
```

## 03. 루팅 탐지 우회

- 디컴파일 후, Jadx에서 봤던 패키지이름과 같은 경로

C:\Users\PC\Downloads\APK Easy Tool v1.60 Portable\1-Decompiled APKs\Uncrackable1\smali\sg\vantagepoint\uncrackable1\				
이름	수정한 날짜	유형	크기	
a.smali	2024-11-16 오후 11:04	SMALI 파일	3KB	
MainActivity\$1.smali	2024-11-16 오후 11:04	SMALI 파일	2KB	
MainActivity\$2.smali	2024-11-16 오후 11:04	SMALI 파일	2KB	
MainActivity.smali	2024-11-16 오후 11:04	SMALI 파일	5KB	

sg/vantagepoint/uncrackable1/

## 03. 루팅 탐지 우회

- MainActivity.smali

```
protected void onCreate(Bundle bundle) {
    if (c.a() || c.b() || c.c()) {
        a("Root detected!");
    }
    if (b.a(getApplicationContext())) {
        a("App is debuggable!");
    }
    super.onCreate(bundle);
    setContentView(R.layout.activity_main);
}
```

Java

```
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 1

    # 루트 탐지를 위한 a() 메소드 호출
    invoke-static {}, Lsg/vantagepoint/a/c; ->a()Z

    # 결과를 v0 레지스터에 저장
    move-result v0

    # 만약 a()에서 루트를 탐지했다면(True), cond_0로 이동
    if-nez v0, :cond_0

    # a()가 거짓이면 b() 메소드 호출
    invoke-static {}, Lsg/vantagepoint/a/c; ->b()Z

    # 결과를 v0 레지스터에 저장
    move-result v0

    # 만약 b()에서 루트를 탐지했다면(True), cond_0로 이동
    if-nez v0, :cond_0

    # b()가 거짓이면 c() 메소드 호출
    invoke-static {}, Lsg/vantagepoint/a/c; ->c()Z

    # 결과를 v0 레지스터에 저장
    move-result v0

    # 만약 c()에서 루트를 탐지하지 않았다면(False), cond_1로 이동
    if-eqz v0, :cond_1

    :cond_0
    # 루트 탐지 메시지
    const-string v0, "Root detected!"

    # 루트 탐지 메시지를 MainActivity의 a 메소드에 전달
    invoke-direct {p0, v0}, Lsg/vantagepoint/uncrackable1/MainActivity; ->a(Ljava/lang/String;)V
```



## 03. 루팅 탐지 우회

- MainActivity.smali

```
protected void onCreate(Bundle bundle) {
    if (c.a() || c.b() || c.c()) {
        a("Root detected!");
    }
    if (b.a(getApplicationContext())) {
        a("App is debuggable!");
    }
    super.onCreate(bundle);
    setContentView(R.layout.activity_main);
}
```

Java

```
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 1

    # 루트 탐지를 위한 a() 메소드 호출
    invoke-static {}, Lsg/vantagepoint/a/c; ->a()Z

    # 결과를 v0 레지스터에 저장
    move-result v0

    # 만약 a()에서 루트를 탐지했다면(True), cond_0로 이동
    if-nez v0, :cond_0

    # a()가 거짓이면 b() 메소드 호출
    invoke-static {}, Lsg/vantagepoint/a/c; ->b()Z

    # 결과를 v0 레지스터에 저장
    move-result v0

    # 만약 b()에서 루트를 탐지했다면(True), cond_0로 이동
    if-nez v0, :cond_0

    # b()가 거짓이면 c() 메소드 호출
    invoke-static {}, Lsg/vantagepoint/a/c; ->c()Z

    # 결과를 v0 레지스터에 저장
    move-result v0

    # 만약 c()에서 루트를 탐지하지 않았다면(False), cond_1로 이동
    if-eqz v0, :cond_1

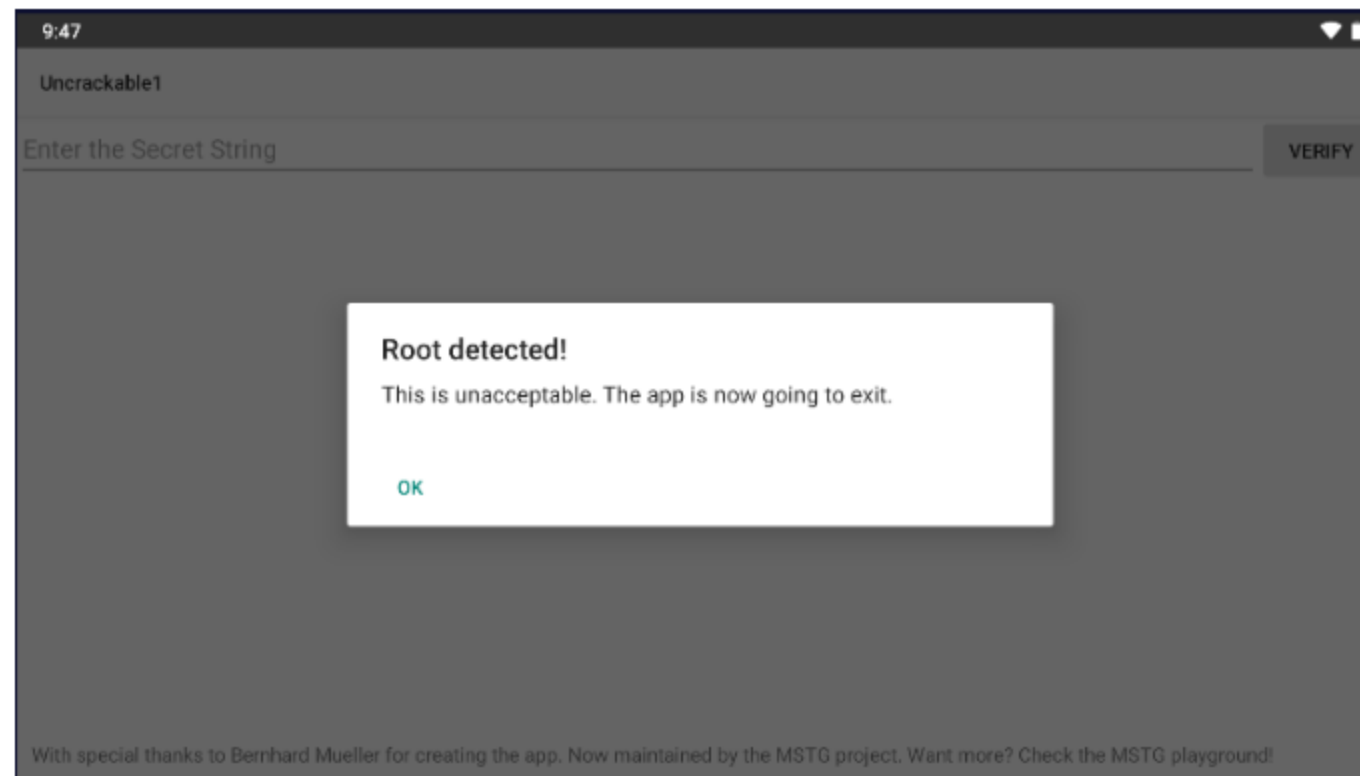
    :cond_0
    # 루트 탐지 메시지
    const-string v0, "Root detected!"

    # 루트 탐지 메시지를 MainActivity의 a 메소드에 전달
    invoke-direct {p0, v0}, Lsg/vantagepoint/uncrackable1/MainActivity; ->a(Ljava/lang/String;)V
```

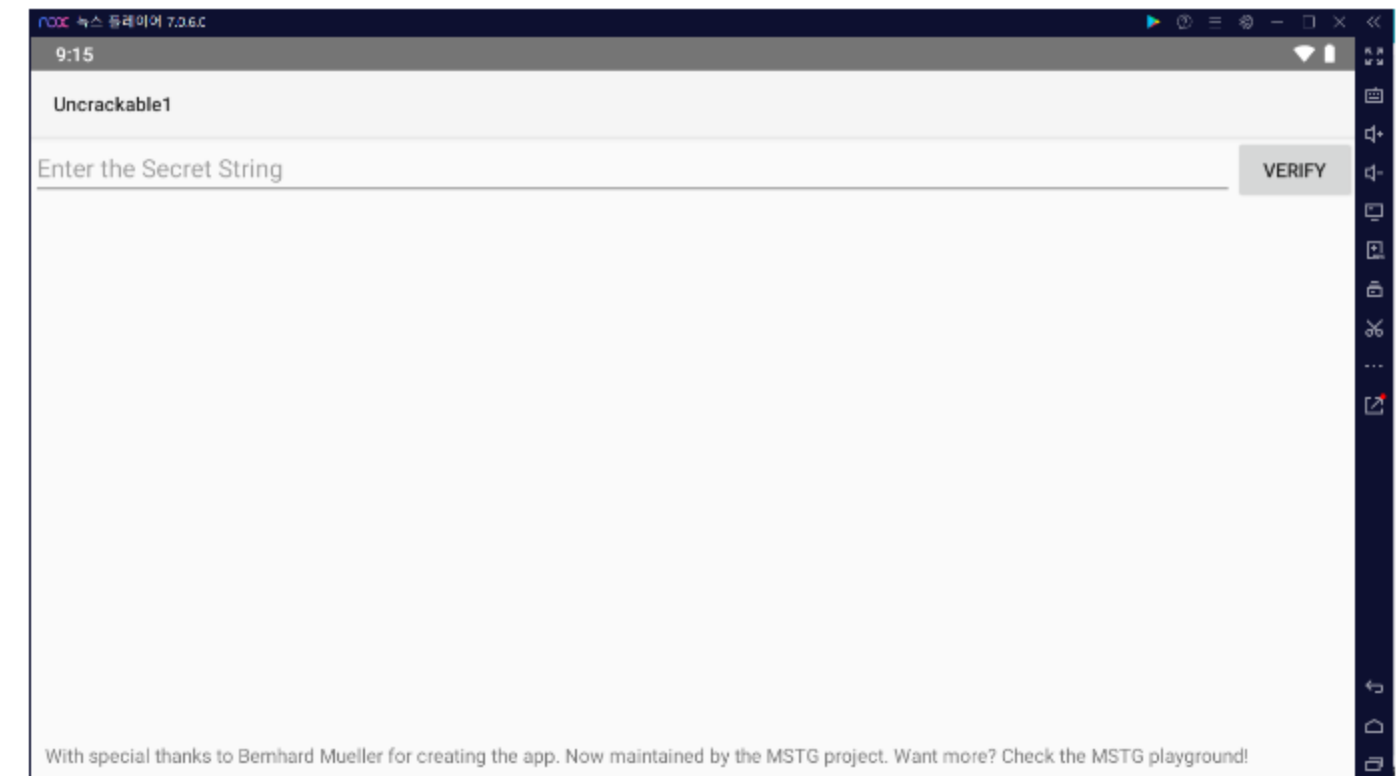
# 리패키징...

## 03. 루팅 탐지 우회

- APK Easy Tool로 리패키징 후 변조 앱 실행



변조 전



변조 후



## 03. 루팅 탐지 우회

- 변조 앱 jadx로 디컴파일

```
protected void onCreate(Bundle bundle) {  
    if (c.a() || c.b() || c.c()) {  
        a("Root detected!");  
    }  
    if (b.a(getApplicationContext())) {  
        a("App is debuggable!");  
    }  
    super.onCreate(bundle);  
    setContentView(R.layout.activity_main);  
}
```

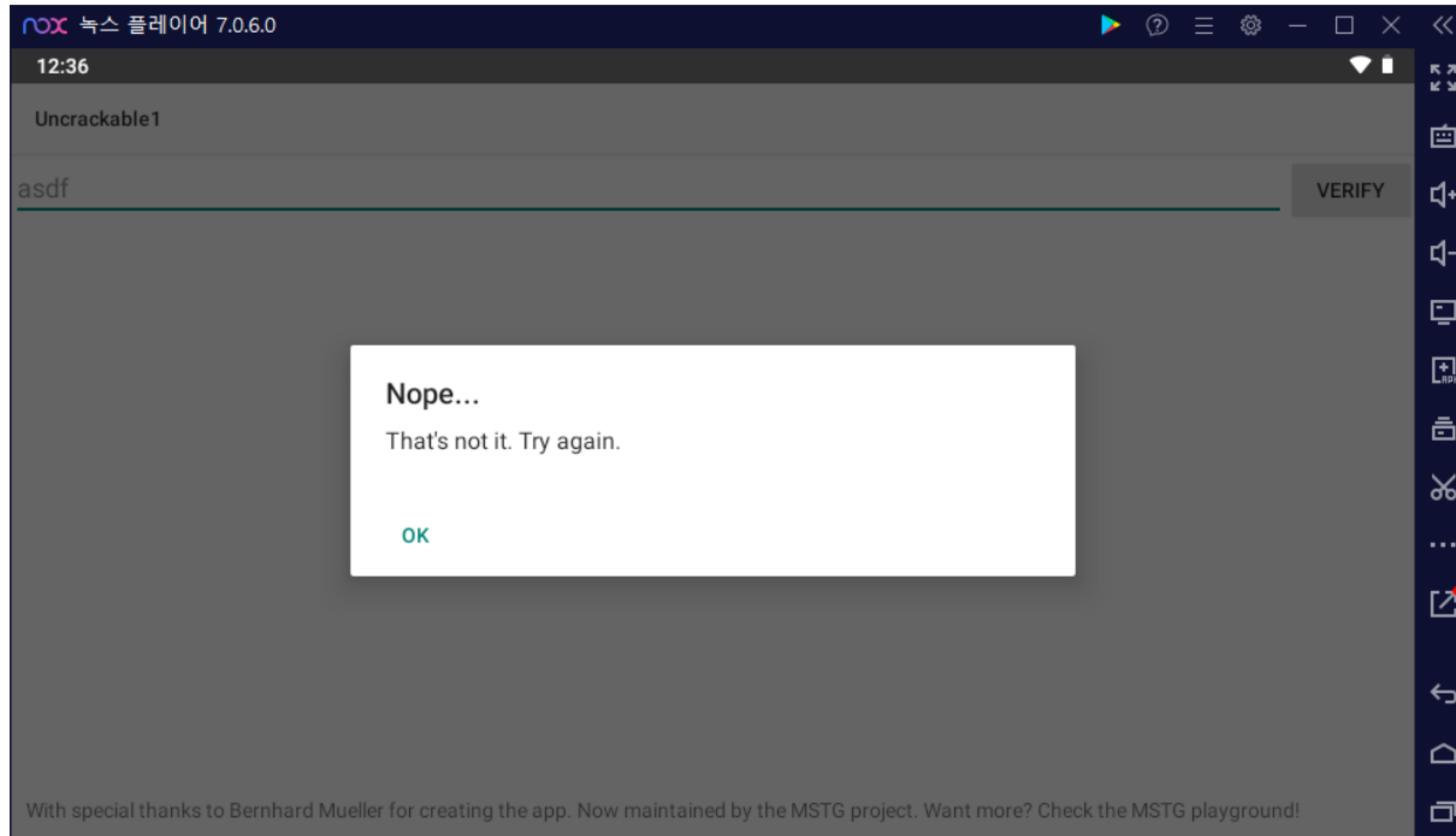
변조 전



```
protected void onCreate(Bundle bundle) {  
    if (b.a(getApplicationContext())) {  
        a("App is debuggable!");  
    }  
    super.onCreate(bundle);  
    setContentView(R.layout.activity_main);  
}
```

변조 후

## 03. Secret String 찾기



## 03. Secret String 찾기

- verify 메소드

```
public void verify(View view) {
    String str;
    String obj = ((EditText) findViewById(R.id.edit_text)).getText().toString();
    AlertDialog create = new AlertDialog.Builder(this).create();
    if (a.a(obj)) {
        create.setTitle("Success!");
        str = "This is the correct secret.";
    } else {
        create.setTitle("Nope...");
        str = "That's not it. Try again.";
    }
    create.setMessage(str);
    create.setButton(-3, "OK", new DialogInterface.OnClickListener() { // from class: sg.vant
        @Override // android.content.DialogInterface.OnClickListener
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
        }
    });
    create.show();
}
```

## 03. Secret String 찾기

- 복호화 후 입력값과 비교

```
public class a {  
    public static boolean a(String str) {  
        byte[] bArr;  
        byte[] bArr2 = new byte[0];  
        try {  
            bArr = sg.vantagepoint.a.a.a(b("8d127684cbc37c17616d806cf50473cc"), Base64.decode("5UJiFctbmgbDoLXmpL12mkno8HT4Lv8dlat8FxR2G0c=", 0));  
        } catch (Exception e) {  
            Log.d("CodeCheck", "AES error:" + e.getMessage());  
            bArr = bArr2;  
        }  
        return str.equals(new String(bArr));  
    }  
  
    public static byte[] b(String str) {  
        int length = str.length();  
        byte[] bArr = new byte[length / 2];  
        for (int i = 0; i < length; i += 2) {  
            bArr[i / 2] = (byte) ((Character.digit(str.charAt(i), 16) << 4) + Character.digit(str.charAt(i + 1), 16));  
        }  
        return bArr;  
    }  
}
```

문자열로 변환된 secret string 값 로그에 출력

## 03. Secret String 찾기

- 복호화 후 입력값과 비교

```
.method public static a([B[B
    .locals 2

    new-instance v0, Ljavax/crypto/spec/SecretKeySpec;

    const-string v1, "AES/ECB/PKCS7Padding"

    invoke-direct {v0, p0, v1}, Ljavax/crypto/spec/SecretKeySpec;-><init>([Ljava/lang/String;)V

    const-string p0, "AES"

    invoke-static {p0}, Ljavax/crypto/Cipher;->getInstance(Ljava/lang/String;)Ljavax/crypto/Cipher;

    move-result-object p0

    const/4 v1, 0x2

    # 0x2는 복호화 모드를 의미 (Cipher.DECRYPT_MODE)
    const/4 v1, 0x2 # v1에 복호화 모드(2)를 설정

    invoke-virtual {p0, v1, v0}, Ljavax/crypto/Cipher;->init(ILjava/security/Key;)V

    invoke-virtual {p0, p1}, Ljavax/crypto/Cipher;->doFinal([B)[B

    move-result-object p0 # 복호화된 결과를 p0에 할당

    # 복호화된 결과(p0)를 반환
    return-object p0
.end method
```

```
invoke-virtual {p0, v1, v0}, Ljavax/crypto/Cipher;->init(ILjava/security/Key;)V

invoke-virtual {p0, p1}, Ljavax/crypto/Cipher;->doFinal([B)[B

move-result-object p0 # 복호화된 결과를 p0에 할당

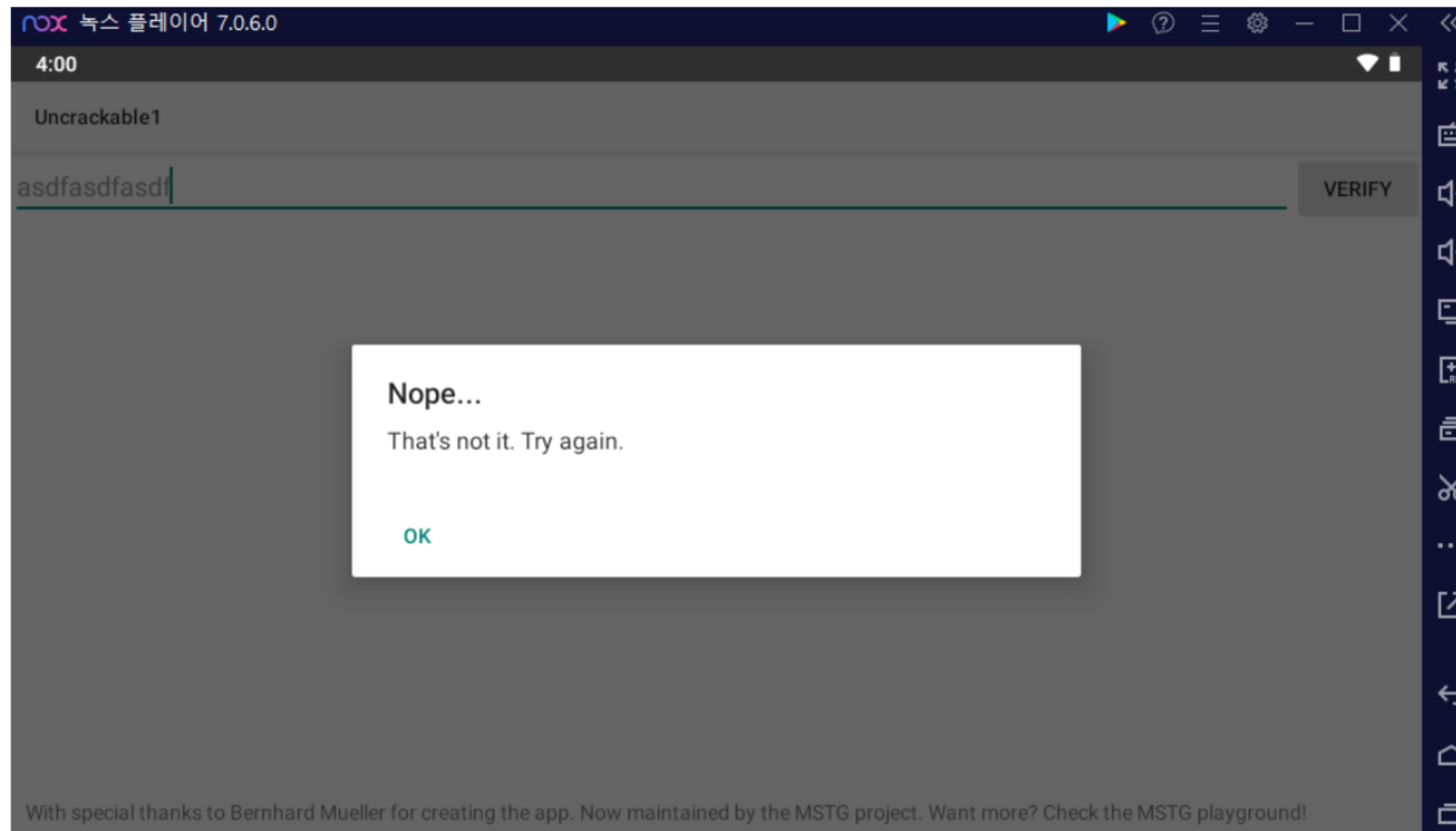
# 복호화된 결과(p0)를 반환
return-object p0
```

```
move-result-object v0
const-string v1, "-----"
new-instance v2, Ljava/lang/String;
invoke-direct {v2, v0}, Ljava/lang/String;-><init>(B)V
invoke-static {v1, v2}, Landroid/util/Log;
->d(Ljava/lang/String;Ljava/lang/String;)I
return-object v0
```

# 리패키징...

## 03. Secret String 찾기

- 리패키징된 앱 실행, secret값에 아무 값이나 입력





## 03. Secret String 찾기

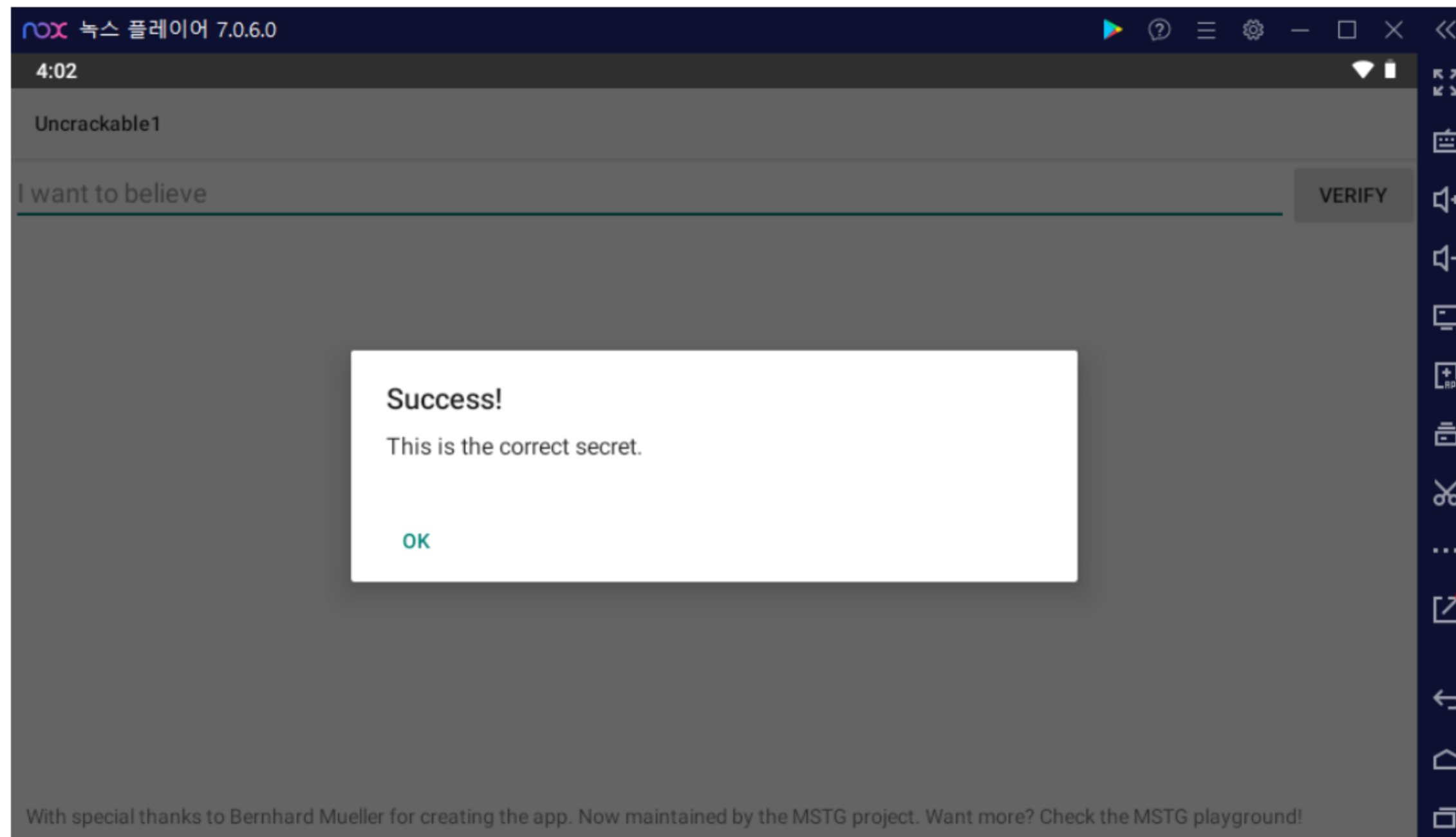
- Logcat으로 Secret String 확인

```
15:59:15.854 9409 9438 E libEGL : called unimplemented OpenGL ES API
15:59:15.930 9409 9409 D -----: I want to believe
15:59:15.941 1554 4550 E : pcm_open called create stream faild
15:59:15.942 1568 1701 D AudioFlinger: mixer(0xeac83e80) throttle end: throttle time(79)
15:59:16.062 1570 1675 D gralloc_nox: gralloc_alloc: Creating ashmem region of size 602112
```



## 03. Secret String 찾기

- Secret String 입력



**Q & A**