



three.apk

apk 파일이 주어진다.



안드로이드 파일에서 실행시켜 주면 값을 입력받는 창이 하나 뜬다.

```
> apktool d three.apk -o three_src
```

apktool을 통해 디컴파일 시켜준다.

AndroidManifest.xml	2025-12-12 오전 12:44	Microsoft Edge H...	1KB
apktool.yml	2025-12-12 오전 12:44	Yaml 원본 파일	2KB
lib	2025-12-12 오전 12:44	파일 풀더	
original	2025-12-12 오전 12:44	파일 풀더	
res	2025-12-12 오전 12:44	파일 풀더	
smali	2025-12-12 오전 12:44	파일 풀더	
unknown	2025-12-12 오전 12:44	파일 풀더	

디컴파일이 되면 apk 파일의 내부 소스를 볼 수 있다.

```
@Override // androidx.appcompat.app.AppCompatActivity, androidx.fragment.a
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    this.text_top = (TextView) findViewById(R.id.text_top);
    this.text_bottom = (TextView) findViewById(R.id.text_bottom);
    this.text_input = (EditText) findViewById(R.id.text_input);
    this.ctx = getApplicationContext();
    System.loadLibrary("hellojni");
    this.text_top.setText(R.string.hint);
}

public void buttonClick(View view) {
    String content = this.text_input.getText().toString();
    this.text_bottom.setText(FlagstaffHill.getFlag(content, this.ctx));
}
```

jadx로 코드를 살펴보면 앱이 실행되면 onCreate가 실행된다.

text_input에 사용자 입력이 저장되고, 버튼을 누르면 buttonClick 함수가 실행되고, 사용자 입력을 getFlag 함수에 전달한다. 그리고 text_bottom에 그 결과를 출력한다.

```
/* loaded from: classes.dex */
public class FlagstaffHill {
    public static native String cilantro(String str);

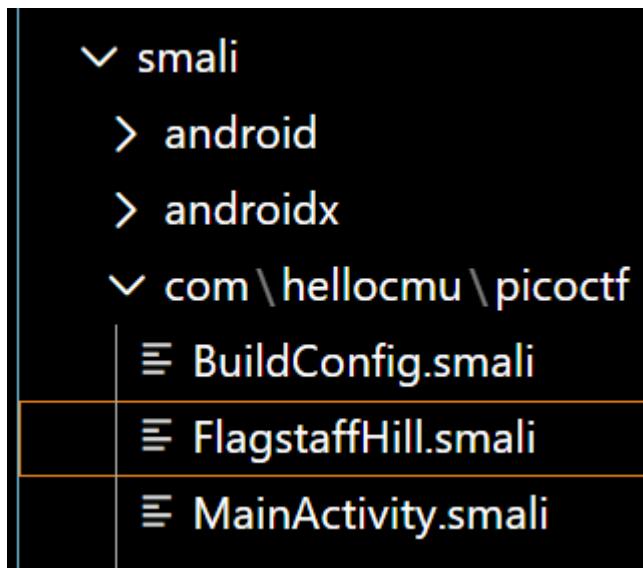
    public static String nope(String input) {
        return "don't wanna";
    }

    public static String yep(String input) {
        return cilantro(input);
    }

    public static String getFlag(String input, Context ctx) {
        String flag = nope(input);
        return flag;
    }
}
```

FlagstaffHill 클래스의 getFlag함수를 보면 무조건 nope 함수로 전달한다. 즉, 어떤 값을 입력해도 "don't wanna"가 출력된다. FLAG를 출력시키려면 아래 yep 함수로 전달해야 하는 것 같다. 아까 디컴파일 시킨 apk 파일의 소스코드를 nope -> yep으로 바꿔주면 될 것 같다.

smali 폴더에 FlagstaffHill 파일이 보인다.



```
.method public static getFlag(Ljava/lang/String;Landroid/content/Context;)Ljava/lang/String;
    .locals 1
    .param p0, "input"    # Ljava/lang/String;
    .param p1, "ctx"      # Landroid/content/Context;

    .line 19
    invoke-static {p0}, Lcom/hellocmu/picoctf/FlagstaffHill;->nope(Ljava/lang/String;)Ljava/lang/String;

    move-result-object v0

    .line 20
    .local v0, "flag":Ljava/lang/String;
    return-object v0
.end method
```

getFlag 함수가 보이고 아까 코드에서 보았던 nope 호출 부분이 보인다. 해당 부분을 yep으로 바꿔준다.

```
.method public static getFlag(Ljava/lang/String;Landroid/content/Context;)Ljava/lang/String;
    .locals 1
    .param p0, "input"    # Ljava/lang/String;
    .param p1, "ctx"      # Landroid/content/Context;

    .line 19
    invoke-static {p0}, Lcom/hellocmu/picoctf/FlagstaffHill;->yep(Ljava/lang/String;)Ljava/lang/String;

    move-result-object v0

    .line 20
    .local v0, "flag":Ljava/lang/String;
    return-object v0
.end method
```

```
> apktool b three_src -o new_app.apk
```

해당 명령을 통해 수정된 패키지를 재빌드 해준다.



새 apk 파일을 생성해준다. 그런데 이 상태로는 apk 설치를 못한다. 왜냐하면 서명을 안해주었기 때문이다.

```
keytool -genkey -v -keystore mykey.keystore -alias myalias -keyalg RSA -keysize 2048 -validity 10000
```

keystore를 생성해주고, jarsigner을 통해 서명을 해주면 된다.

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore mykey.keystore new_app.apk myalias
```

재빌드한 apk 파일을 에뮬레이터에 업로드하고 실행시켜주면 플래그가 나온다.

