API Key 탈취 & 재배포

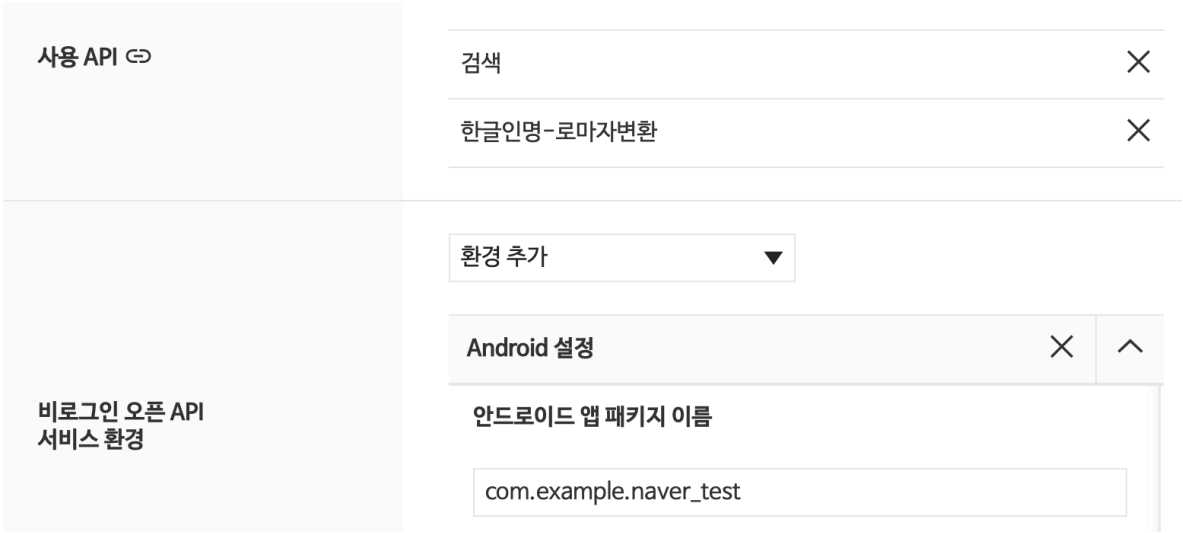
1. Naver API (검색, 로마자표기)
2. Kakao API (카카오 로그인). // TODO
3. FaceBook API (Facebook Login)
4. Amazon API (Amazon Login)

==========

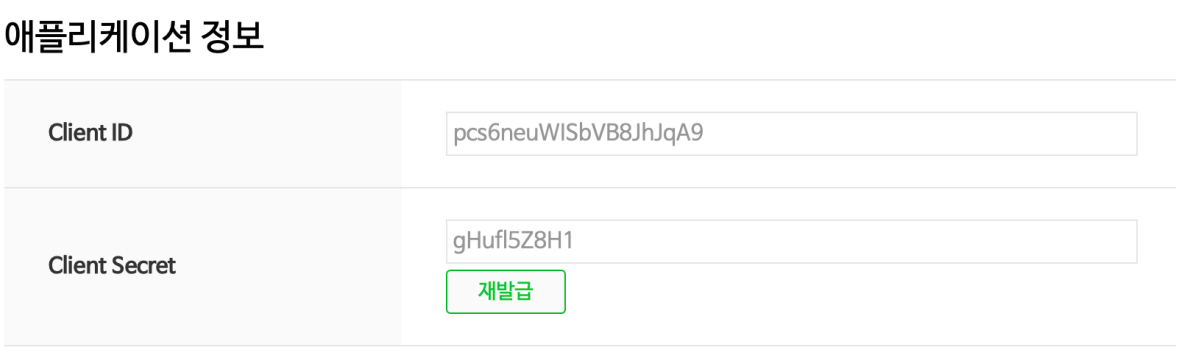
[1] Naver API (검색, 로마자 표기)

1. https://hyongdoc.tistory.com/167
2. https://titanic1997.tistory.com/36
3. https://developers.naver.com/products/search

- Naver API를 사용하기위해 Key를 발급받기 위해 요구되는 정보



- 발급된 API Key



- 발급된 Key를 이용해 API를 사용하는 코드 작성

```java
String clientId = "YOUR_CLIENT_ID";//애플리케이션 클라이언트 아이디값";
String clientSecret = "YOUR_CLIENT_SECRET";//애플리케이션 클라이언트 시크릿값";
try {
    String text = URLEncoder.encode("홍길동", "UTF-8");
    String apiURL = "https://openapi.naver.com/v1/krdict/romanization?query="+ text;
    URL url = new URL(apiURL);
    HttpURLConnection con = (HttpURLConnection)url.openConnection();
    con.setRequestMethod("GET");
    con.setRequestProperty("X-Naver-Client-Id", clientId);
    con.setRequestProperty("X-Naver-Client-Secret", clientSecret);
```

- con.setRequestProperty( ) 함수를 통해 발급된 API Key인 “Client ID”, “Client Secret” 값을 String형으로 사용함

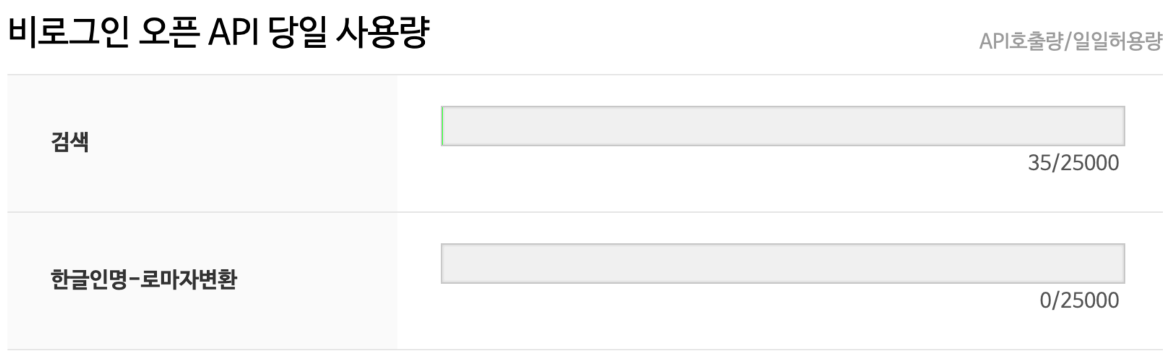
//
//

- ByteCode의 동적 분석, 정적 분석을 통해 해당 Key값 추출할 수 있음.
- (Byte DBI를 이용해 동적으로 스트링을 관찰)

```
44640
44641     [0] const-string v0, "pcs6neuWISbVB8JhJqA9" // string@1940// (cur) void com.example.naver_test.MainActivity.searchNaver(java.lang.String)
44642     [2] const-string v1, "gHufl5Z8H1" // string@1730// (cur) void com.example.naver_test.MainActivity.searchNaver(java.lang.String)
44643     [4] const/4 v2, #+5// (cur) void com.example.naver_test.MainActivity.searchNaver(java.lang.String)
```

- 추출한 API Key를 다른 App을 제작할때 사용하면, 별도의 등록 절차 없이 API를 사용할 수 있음.

- 탈취한 API Key를 이용해 API를 실행 시켰을때, API Key 발급시 등록한 앱이 실행된것처럼, API 사용량이 증가함.



===

[2] Kakao API (kakao Login)

- // TODO

===

[3] FaceBook API (Facebook Login)

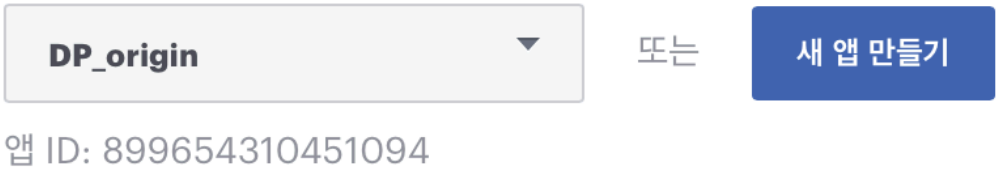
1. https://developers.facebook.com/docs/facebook-login/android
2. https://knowledge.digicert.com/solution/SO17389.html

- FaceBook API를 사용하기 위해 요구되는 정보

- FaceBook Developer에서 새로운 프로젝트를 선택
- 하단에 존재하는 “앱 ID” 가 사실상 API Key의 역할을 함

1. 앱 선택 또는 새 앱 만들기

앱을 선택하거나 새로 만들어 다음 픽셀 코드로 앱 정보를 가져오세요.



앱 ID: 899654310451094

- FaceBook API를 사용하여 앱을 구현하기위해 필요한 작업
- /app/res/vlaues/strings.xml 에 “앱 ID”와 “fb” + “앱 ID” 2가지를 등록
  ex) 899654310451094 , fb899654310451094
- Meta-data에서 ‘/app/res/vlaues/strings.xml’ 에 등록된 ID를 사용

4. 리소스 및 메니페스트 수정

회원님의 Facebook 앱 ID 및 Chrome 맞춤 탭을 활성화하기 위해 필요한 Facebook 앱 ID의 문자열을 만듭니다. 또한 `FacebookActivity`를 Android 매니페스트에 추가합니다.

1. `/app/res/values/strings.xml` 파일을 엽니다.
2. 다음을 추가합니다.

```
<string name="facebook_app_id">899654310451094</string> <string name="fb_login_protocol_s
```

3. `/app/manifest/AndroidManifest.xml` 파일을 엽니다.
4. `application` 요소 뒤에 다음 `uses-permission` 요소를 추가합니다.

```
  <uses-permission android:name="android.permission.INTERNET"/>
```

5. 다음 `meta-data` 요소, Facebook에 대한 활동, Chrome 맞춤 탭에 대한 활동 및 인텐트 필터를 `application` 요소 내에 추가합니다.

```
<meta-data android:name="com.facebook.sdk.ApplicationId" android:value="@string/facebook_
```

- 해당 Facebook Project를 사용할 애플리케이션과의 연결
  (패키지이름, 기본 액티비티 클래스 등록)

5. 패키지 이름 및 기본 클래스를 앱과 연결

- Facebook API의 경우 패키지의 등록과 별개로 키에 의한 인증 또한 진행한다.
- 인증키를 등록하는 방식은 2가지가 있다.

  1) 개발자 키의 경우 keytool을 이용해 제작하기 때문에 password만 지정하여 제작하면 됨

  $ keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore | openssl sha1 -binary | openssl base64

  2) 릴리즈 키의 경우 Signed APK를 생성할 때 사용되는 키를 기반으로 해시값을 구함

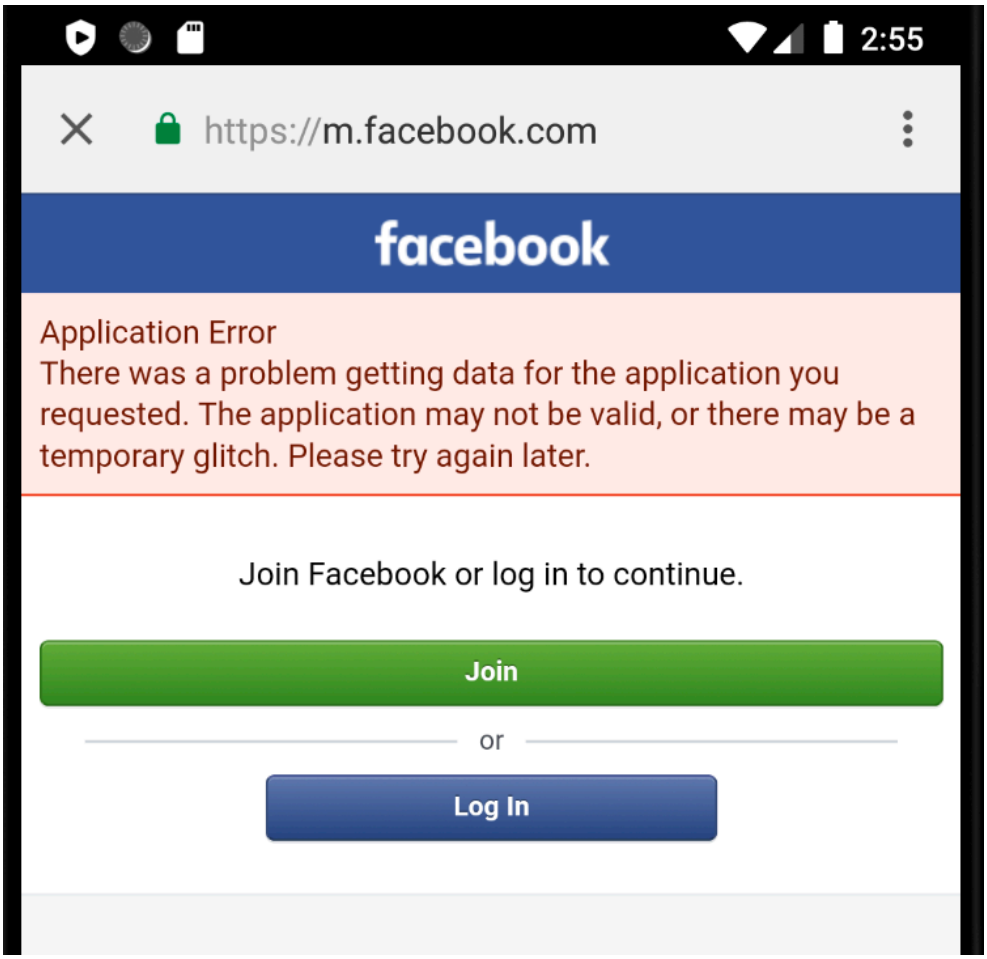
  ** 최근 산업계에서는 jks가 아닌 pkcs12형식을 사용하기 때문에 Android Studio 생성된 jks 키를 pkcs12형식으로 바꾸어 생성해야함.

```
keytool -importkeystore -srckeystore [MY_KEYSTORE.jks] -
destkeystore [MY_FILE.p12] -srcstoretype JKS - deststoretype
PKCS12 -deststorepass [PASSWORD_PKCS12]
```

List of example parameters:

MY_KEYSTORE.jks: path to the keystore that you want to convert.
MY_FILE.p12: path to the PKCS#12 file (.p12 or .pfx extension) that is going to be created.
PASSWORD_PKCS12: password that will be requested at the PKCS#12 file opening.

- $ keytool -exportcert -alias [YOUR_RELEASE_KEY_ALIAS] -keystore [YOUR_RELEASE_KEY_PATH] | openssl sha1 -binary | openssl base64
- “개발자 키 해시”, “릴리즈 키 해시” 중 원하는 방법을 선택해, 해시값을 계산하여 해시 키를



등록한다.

키 해시

ga0RGNYHvNM5d0SLGQfpQWAPGJ8= × OqtvFps5WZ1wU19pAJrSnHBFrgQ= ×

- 앞에서 부터 "개발자 키 해시", "릴리즈 키 해시" 2가지를 모두 등록한 모습이다.

- 실질적으로 App을 생성할때 필요한 정보는 "앱 ID" 뿐임, 잘못된 "앱 ID"를 사용할 경우 아래와 같은 오류가 발생함

```
E/GraphResponse: {HttpStatus: 404, errorCode: 803, subErrorCode: -1, errorType: OAuthException, errorMessage: (#803) Some of the aliases you requested do not exist: 009654310451094}
E/GraphResponse: {HttpStatus: 404, errorCode: 803, subErrorCode: -1, errorType: OAuthException, errorMessage: (#803) Some of the aliases you requested do not exist: 009654310451094}
E/GraphResponse: {HttpStatus: 404, errorCode: 803, subErrorCode: -1, errorType: OAuthException, errorMessage: (#803) Some of the aliases you requested do not exist: 009654310451094}
E/GraphResponse: {HttpStatus: 404, errorCode: 803, subErrorCode: -1, errorType: OAuthException, errorMessage: (#803) Some of the aliases you requested do not exist: 009654310451094}
```

- 오류의 형태가 HttpStatus이고, request do not exist 라는 것을 보아 앱 ID가 포함된 URL로 접근하는것을 볼 수 있음.

- 오류가 발생한 Request를 통해 로딩되는 페이지는 아래와 같다.

Application Error
There was a problem getting data for the application you requested. The application may not be valid, or there may be a temporary glitch. Please try again later.

Join Facebook or log in to continue.

- 위 페이지가 로딩될때 사용되는 URL은 아래와 같다.

- client_id, redirect_uri 의 섹션에, 앱을 제작시 strings.xml에 입력한 값들이 그대로 포함되어 있음을 볼 수 있음

- 009654310451094, fb009654310451094 라는 임의의 값(잘못된 “앱ID”)을 사용하여 애플리케이션을 생성한 애플리케이션에서 생성된 결과.

https://m.facebook.com/v5.0/dialog/oauth?client_id=009654310451094&cbt=1580839069219&e2e=%7B"init"%3A1580839069219%7D&ies=1&sdk=android-5.15.3&sso=chrome_custom_tab&scope=email&state=%7B"0_auth_logger_id"%3A"4f7b98f3-9be6-4992-b0dc-a46431d1deb3"%2C"3_method"%3A"custom_tab"%2C"7_challenge"%3A"8ae6hhoh3ohbg82roash"%7D&default_audience=friends&redirect_uri=fb009654310451094%3A%2F%2Fauthorize&auth_type=rerequest&response_type=token%2Csigned_request%2Cgraph_domain&return_scopes=true

//
//

- ByteCode의 동적 분석, 정적 분석 모두를 통해 해당 Key값 추출할 수 있음.
- (만들어진 APK를 정적분석 “$apktool d [apk_file]”을 통해 Meta-data 로 사용되는 “앱 ID” 값을 추출할 수 있음)

```
    <string name="facebook_app_id">899654310451094</string>
    <string name="fb_login_protocol_scheme">fb899654310451094</string>
    <string name="search_menu_title">Search</string>
    <string name="status_bar_notification_info_overflow">999+</string>
</resources>
strings.xml                                                    49,19
```

- 추출된 “앱 ID” 를 이용해 App을 제작하여 재배포가 가능함
- 재배포시 사용되는 pkcs 의 hash값은 전혀 영향을 미치지 않음

- $ keytool -exportcert -alias test -keystore /Users/dploy/pkcs12_akey | openssl sha1 -binary | openssl base64
  Enter keystore password: [PASSWORD]
  aGmzOYdi2hSxuhRhpGZ8Dt1ccus=

- 공격자의 App을 Sign 할때 사용된 PKCS 키 역시 원본 앱과 전혀 다른 키와 서명이 사용되었다, 공격자의 pkcs의 키 해시 값은 위와 같다.

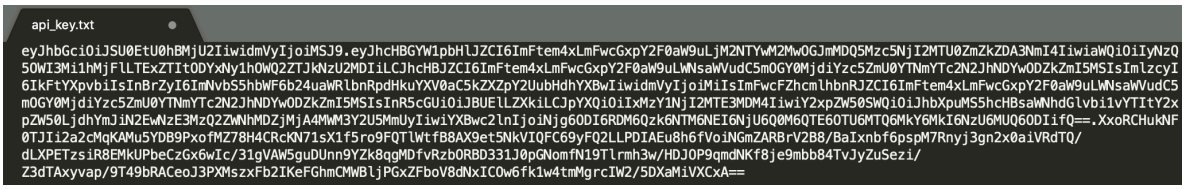
==

[3] Amazon API (Amazon Login)

- Amazon은 Jar 파일 형식의 라이브러리를 제공하며, 해당 라이브러리를 사용함으로써, API를 이용할 수 있게된다.

- Amazon 은 공식 사이트에서 제공하는 sample App을 대상으로 API Key 탈취를 진행함.

- Amazon의 경우 API Key Stream(문자열)을 제공하는 것이 아니라, API_key.txt 라는 파일을 제공해 준다.
- 그 내용은 다음과 같다.



- api_key.txt의 전체 문자열을 “.” 기준으로 분리하면 3개의 부분이 생김
- 3 부분은 차례로 [Header], [Payload], [Signature] 섹션을 의미함
- [Header], [Payload]는 base64(UTF-8)로 디코딩하면 다음과 같음.

```
{
	"alg":"RSA-SHA256",
	"ver":"1"
}

{
	"appFamilyId":"amzn1.application.
365603c08bf049379626154ffdd076b8",
	"id":"27499b72-a21e-11e2-8617-a9d6e2d75602",
	"appId":"amzn1.application-
client.f8f427bc79fe4a3fa767ba46086dfb91",
	"iss":"Amazon",
	"pkg":"com.amazon.identity.auth.device.lwaapp",
	"ver":"2","appVariantId":"amzn1.application-
client.f8f427bc79fe4a3fa767ba46086dfb91",
	"type":"APIKey",
	"iat":"1365626117038",
	"clientId":"amzn1.application-oa2-client.
7abbb7a0717346eca06c2081c7ce92e2",
	"appsig":"68:82:D3:C9:53:4B:65:CC:A1:95:14:2F:2B:75:1D:82"   //
16byte => MD5
}

// Decoding 하지 않은 Signature


XxoRCHukNF0TJIi2a2cMqKAMu5YDB9PxofMZ78H4CRcKN71sX1f5ro9FQTlWtf
B8AX9et5NkVIQFC69yFQ2LLPDIAEu8h6fVoiNGmZARBrV2B8/
BaIxnbf6pspM7Rnyj3gn2x0aiVRdTQ/
	dLXPETzsiR8EMkUPbeCzGx6wIc/
31gVAW5guDUnn9YZk8qgMDfvRzbORBD331J0pGNomfN19Tlrmh3w/
```

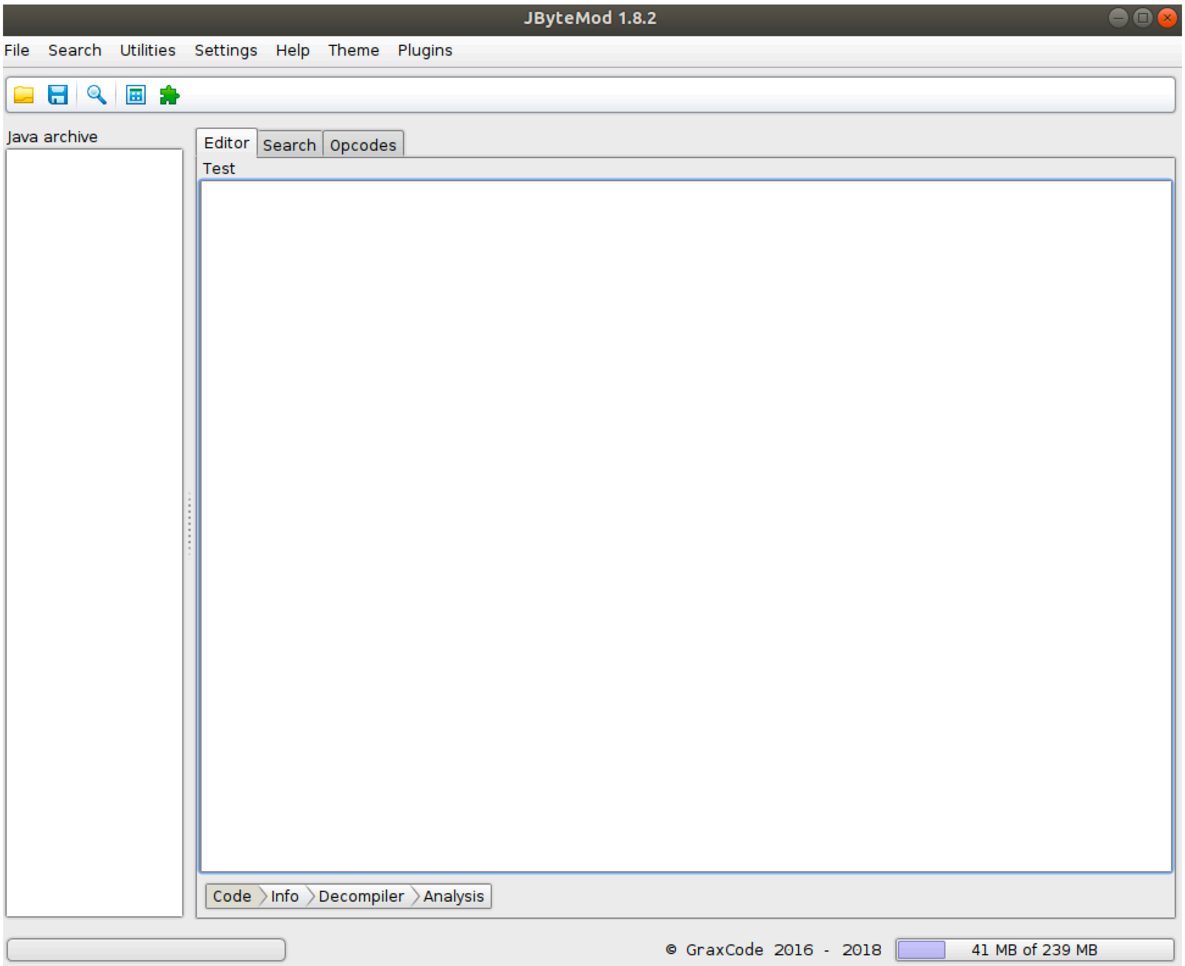
HDJOP9qmdNKf8je9mbb84TvJyZuSezi/Z3dTAxyvap

9T49bRACeoJ3PXMszxFb2lKeFGhmCMWBljPGxZFboV8dNxICOw6fk1w4tmMgr
cIW2/5DXaMiVXCxA==

- Header는 api_key.txt에서 사용하는 인증 방식이 무엇인지를 "alg" 필드를 통해 확인할 수 있다.
- Payload는 해당 Amazon에서 api_key.txt의 값에서 실제 이용하는 데이터들을 담고있다.
- Signature 는 "alg" 필드 값의 인증 방식을 통해 Payload의 veridation check를 하기 위한 데이터로 예측된다.

//

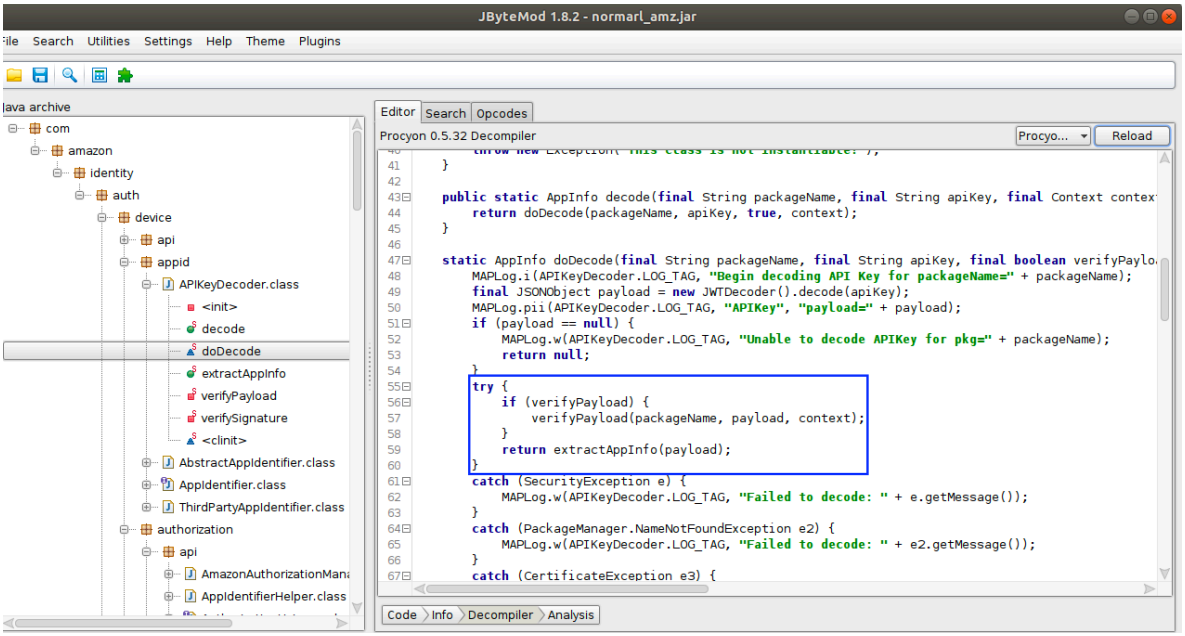
- Amazon API사용을 위해 제공되는 .jar 파일을 아래의 도구를 이용해 패치한다.
- 도구 : JByteMod : Jar, Class 파일을 patch할 수 있음



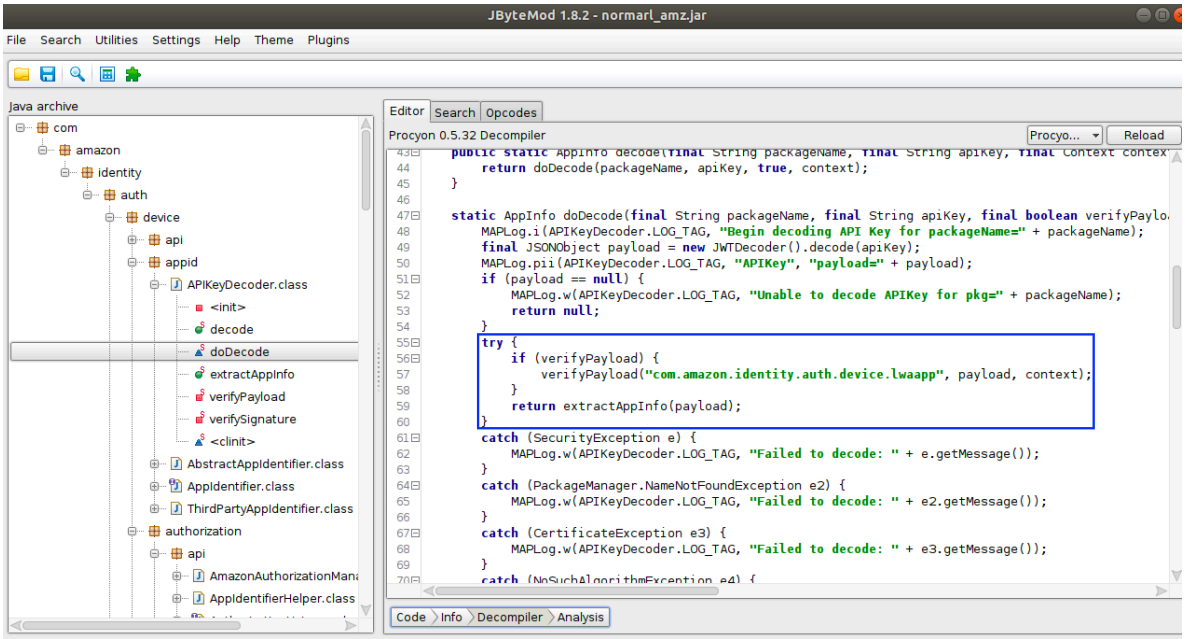
- api_key.txt 파일만 탈취하여 가져온 경우 아래와 같은 에러가 발생

```
W/com.amazon.identity.auth.device.appid.APIKeyDecoder: Failed to decode: Decoding fails: package names don't match! - com.example.amz_test != com.amazon.identity.auth.device.lwaapp
W/com.amazon.identity.auth.device.appid.APIKeyDecoder: Unable to decode APIKey for pkg=com.example.amz_test
```

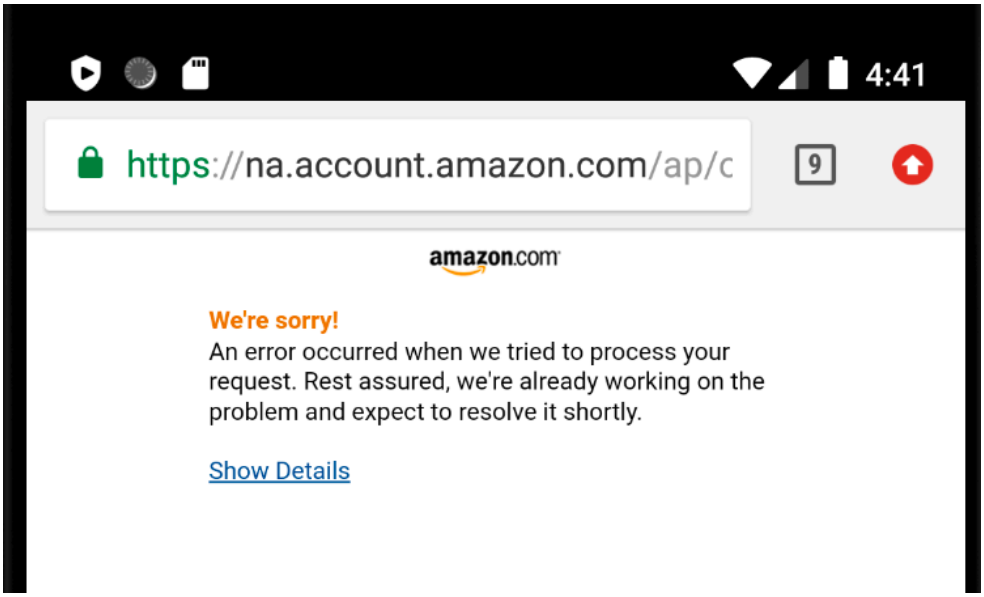
- 에러가 발생하는 이유는 doDecode 함수에서 verifyPayload, verifySignature 작업을 통해 이루어짐
- 에러가 나는 이유가 되는 정보는 api_key.txt 내부의 정보와, 현재 packageName 뿐이다.
- 따라서 해당 함수의 시작 지점에 packageName을 탈취한 정보로 고정시켜 놓는다.



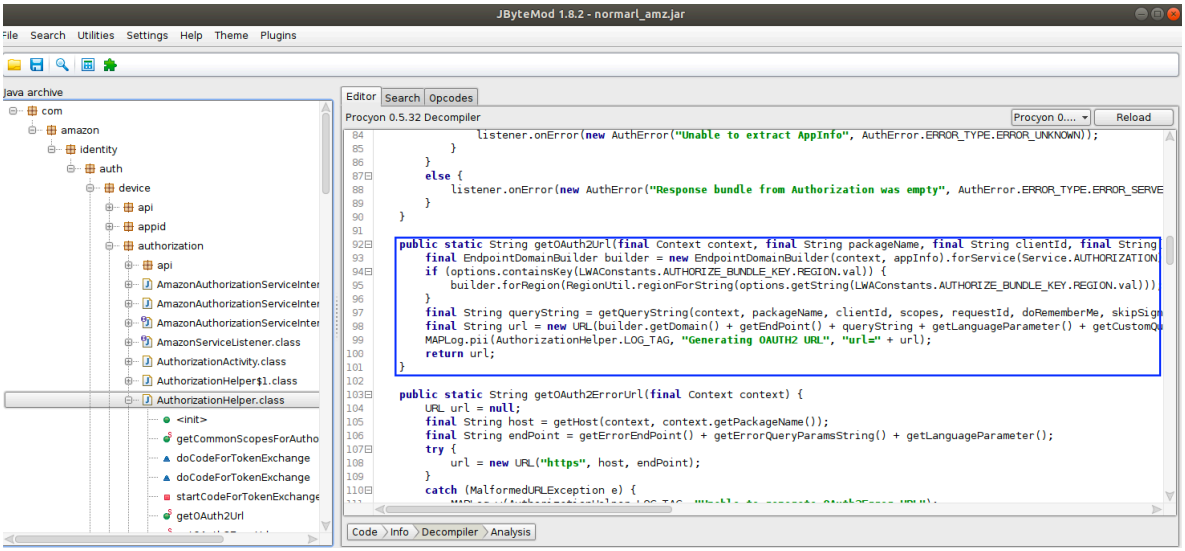
- 패치된 코드



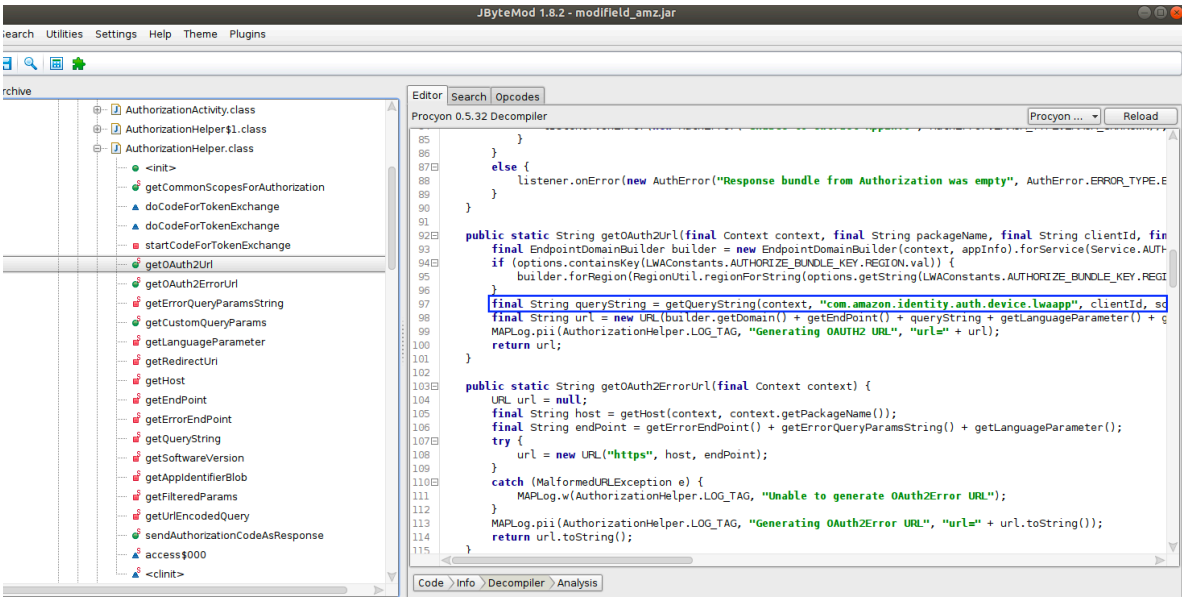
- 패치 후, 앱은 정상 실행이 되지만 올바른 로그인 화면이 팝업되지 않고 아래와 같은 화면이 팝업된다.



- 아래와 같은 로직에 의해 위와 같은 에러가 발생
- 아래의 코드상에서 packageName에 해당하는 정보는 device로 부터 직접 가져온것임.
- 따라서 파라미터에 const String 형태로 문자열을 넣은 식으로 패치를 진행



- 패치된 코드는 아래와 같다.



- 2가지의 로직을 패치를 통해 수정하면 무단 사용이 가능해짐