Hidden Broker - Writeup

1) Find the exposed entrypoint

1. Unpack the APK or inspect manifest to find exported components:
2. unzip -p hiddenbroker.apk AndroidManifest.xml | sed -n '1,200p'
3. Look for exported activities. You should see an exported BrokerActivity (or similar). That tells you there's an entrypoint reachable from outside the app.

What to look for
<activity android:name=".BrokerActivity" android:exported="true">
If exported, it can be started with adb shell am start.

2) Static code analysis (decompile)

1. Open the APK in jadx (or apktool + JADX):
   jadx-gui hiddenbroker.apk — browse classes.
2. Search for keywords: BrokerActivity, SecretChecker, FlagProvider, NativeLib, getHmacSecret, getNativeKey.
3. Read these classes to understand the flow: BrokerActivity reads broker_token extra → calls SecretChecker.verify… → on success calls FlagProvider.getFlag(msg).

Key things to note

- Token format is visible: Base64(message + ":" + signatureHex).
- signature is computed as HMAC_SHA256(message, hmac_secret).hexdigest().upper().
- The HMAC secret and nativeKey are obtained via JNI (NativeLib.getHmacSecret(), NativeLib.getNativeKey()).

3) Locate native library and try static extraction

1. Verify .so is packaged:
2. unzip -l hiddenbroker.apk | grep libbrokerkey
3. Extract the .so for your device ABIs:
4. unzip -j hiddenbroker.apk lib/arm64-v8a/libbrokerkey.so -d ./so
5. Run strings to look for readable secrets:
6. strings so/libbrokerkey.so | grep -i broker || true
7. strings so/libbrokerkey.so | grep -i nAt1v3 || true
8. If strings reveal broker_2025_secret_native or nAt1v3S3cr3t_2025, that's enough to generate tokens / decrypt offline.

If strings are not obvious

Load libbrokerkey.so into Ghidra/IDA and inspect functions Java_com_magical_hiddenbroker_NativeLib_getHmacSecret / ...getNativeKey. The returned string may be constructed or obfuscated — reverse it.

4) Dynamic extraction (alternate method)

If static analysis fails, extract secrets at runtime:

Use Frida to call JNI functions from the running app and print returned strings:

```
// frida-script.js
Java.perform(function () {
```

```
var NativeLib = Java.use('com.magical.hiddenbroker.NativeLib');
console.log('hmac:', NativeLib.getHmacSecret());
console.log('nativeKey:', NativeLib.getNativeKey());
});
```

Run:

```
frida -U -f com.magical.hiddenbroker -l frida-script.js --no-pause
```

- Alternatively use gdb/ptrace or dump process memory and search.

5) Build a valid token (forge HMAC)

Once you have the HMAC secret (hmac_secret) and chosen message (msg), produce the token:

Python generator

```
import base64, hmac, hashlib
key = b"broker_2025_secret_native"   # must match NativeLib.getHmacSecret()
msg = b"trusted_client"              # must match expected message
sig = hmac.new(key, msg, hashlib.sha256).hexdigest().upper()
token = base64.b64encode(msg + b":" + sig.encode()).decode()
print(token)
```

6) Trigger the broker via ADB with the forged token

Run:

```
adb shell am start -n com.magical.hiddenbroker/.BrokerActivity --es broker_token "<TOKEN>"
```

Check results

Flag appears as a popup on the phone!:)