<네트워크보안 실습 과제 (2) - A>

학번 / 학과: B811115 / 컴퓨터공학과
이름: 유병익

<어플리케이션 #1>

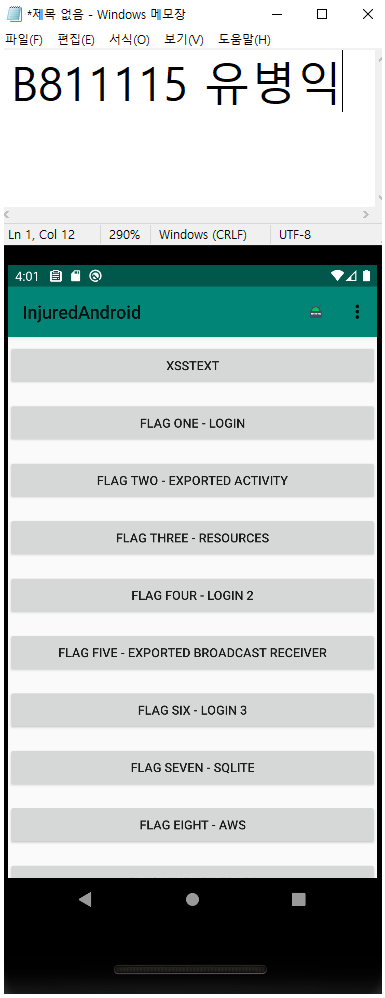
1. 타겟 어플리케이션 설명

Injured Android App은 CTF 형식을 사용해 안드로이드 취약점에 대한 공부에 도움을 줄 수 있도록 제작된 App이다. 해당 App을 통해 이전 실습 3-1에서 Application Patching을 통해 안드로이드 코드를 변경하고, 비정상적인 동작을 수행하도록 APK 변조하는 공격을 재현하고, 해당 취약점을 분석할 예정이다.

2. 취약점에 대한 공격 재현

#Before Patching

-해당 App을 수정하기 전에는 App을 실행할 때, 특이한 문자열 박스가 출력되지 않는다.

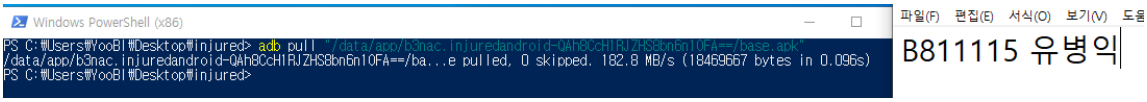


#Location

-adb 프롬프트상에서 pm list packages -f | grep "injured" 명령어를 수행해 App의 설치 경로를 알아내고, adb pull 명령어를 사용해 Apk파일을 작업할 디렉토리로 불러온다.

- pm list packages -f | grep "injured" 수행

-adb pull "/data/app/b3nac.injuredandroid-DmAWFt_h2YLScauxbhrQJA==/base.apk" 수행

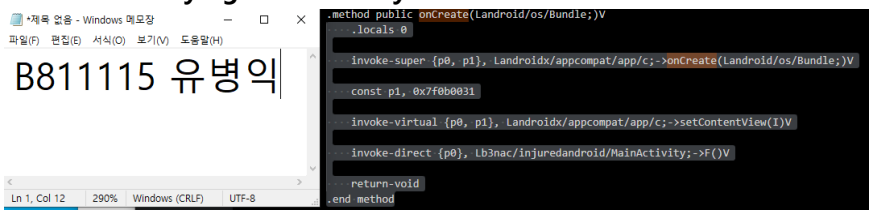


#Decompile & Modifying Smali

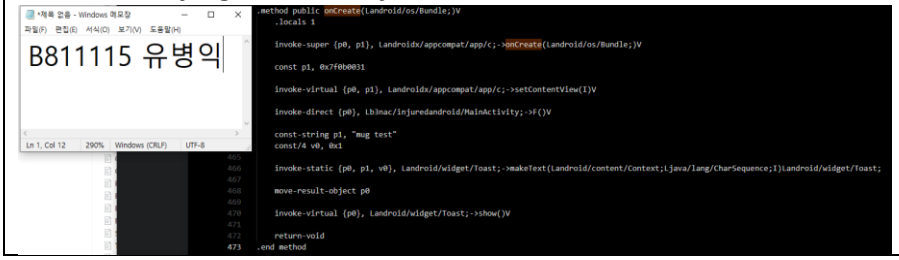
-APK Easy Tool 을 사용해 base.apk를 Decompile한다. Decompile된 파일들이 저장된 폴더의 smali₩b3nac₩injuredandroid 위치의 MainActivity.smali 파일을 코드 에디터(VS Code)를 사용해 수정한다.

-Decompile with APK Easy Toll

-Before Modifying MainActivity.smali



-After ModifyingMainActivity.smali



#Recompile & Reinstall

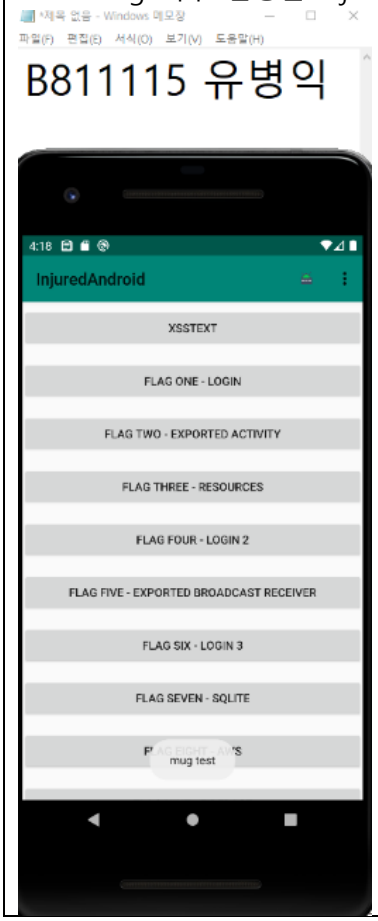
- MainActivity.smali를 수정한 뒤 APK Easy Tool을 활용해 Recompile한다. 이후 AVD 환경에설치되어 있는 Injured Android App을 삭제하고, Recompile된 base.apk를 AVD에 설치한다.

-Recompile

-Reinstall

#After Patching

-Patching 이후 실행된 Injured Android App은 mug test라는 문자열 박스를 출력한다.



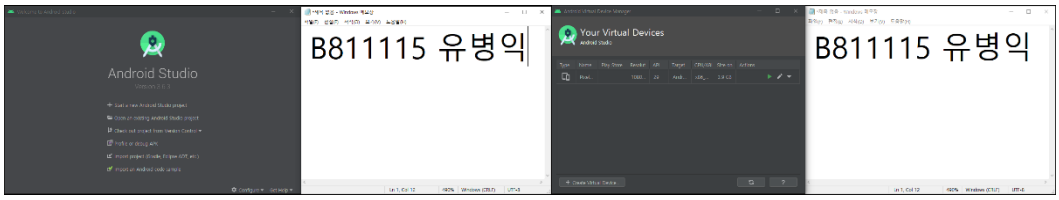
3. 취약점 분석에 사용한 모든 도구 및 분석 방법 설명

#Android Studio

-Android Application 개발을 위해 Jetbrains의 IntelliJ를 기반으로 만든

-AVD 가상 환경에서 App을 설치하고 구동하는 과정을 위해 사용한다.

-Android Studion를 통해 PC환경에서 Application의 구동하고, 특정 작업을 수행하도록 하여 그 과정을 관찰할 수 있다.



#APK Easy Tool

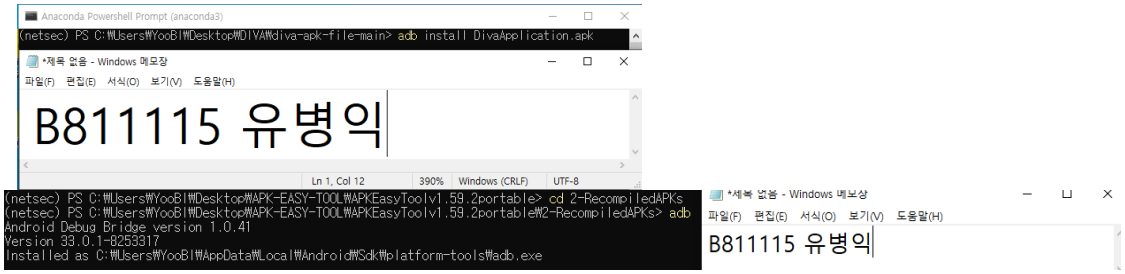
-Apk Easy Tool은 Application의 APK 파일을 관리, 서명, 컴파일 및 디컴파일할 수 있는 GUI Application이다.

-해당 실습에서 Apk Easy Tool을 활용해 Apk파일을 Decompile 하고 수정한 파일을 Recompile 할 때 사용한다

#ADB

- 안드로이드 단말기와 데스크톱 간에 통신을 할 때 필요한 도구.

- adb 명령어를 사용하여 기존 App 설치 및 제거와 변조된 App을 설치하는 작업에 사용한다.

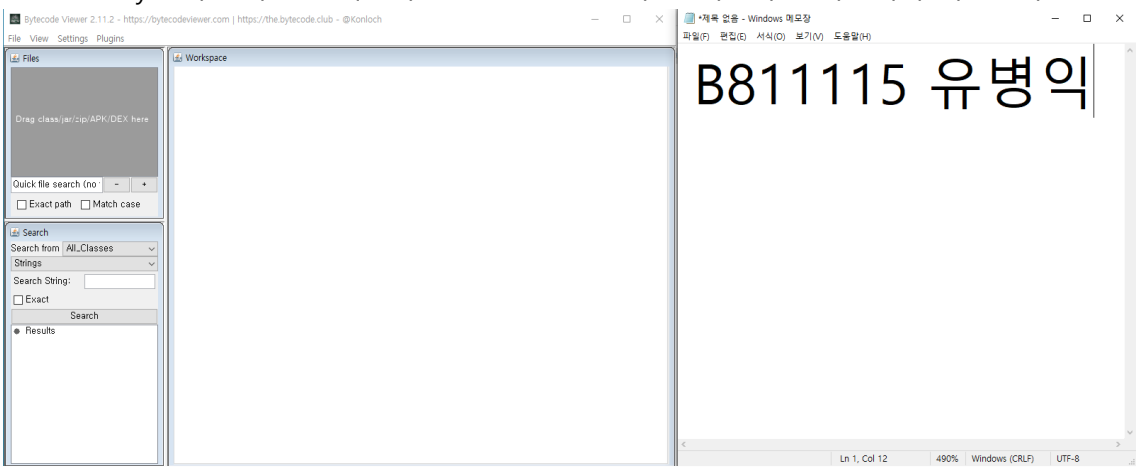


#ByteCodeViewer

-Java, Android 등 소스코드 리버싱 분석 도구이다.

-APK파일을 Decompile하여 바이트코드 형태와 클래스 파일을 자바 파일로 변환하여 소스파일로 복원하여 같이 보여준다.

-MainActivity를 구현하는 클래스의 보면 onCreate함수를 확인하는 과정에서 사용한다.

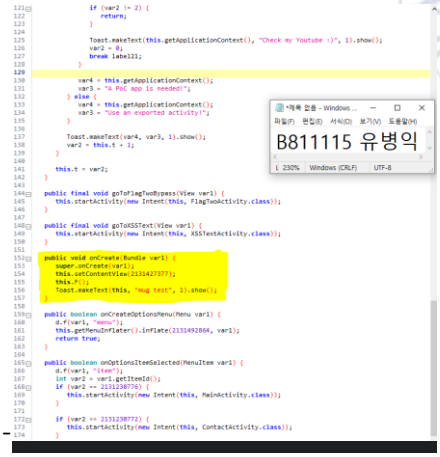


4. 취약점 분석 결과 설명

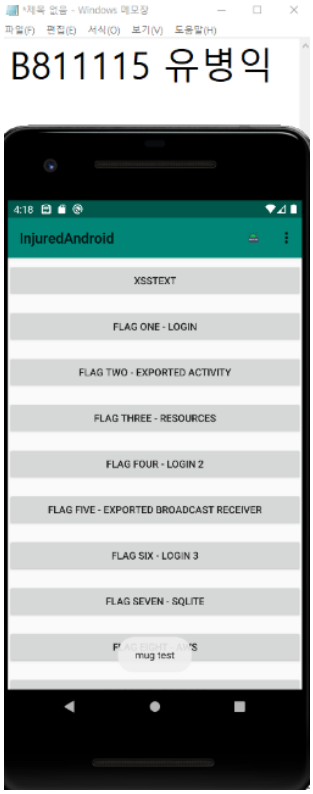
-AndroidManifest.xml 파일에서 action에 해당하는 값이 MAIN인 액티비티를 확인

--ByteCodeViewer를 통해 MainActivity를 구현하는 클래스를 보면 onCreate함수를 확인할 수 있는데 해당 함수는 무조건 Activity가 실행될 때 가장 먼저 호출된다.

-따라서 Oncreate 함수가 변경되면 App이 의도치 않은 동작을 할 수 있다.



-APK Easy Tool 을 사용하여 apk파일을 Decompile하여 만들어진 파일들 중,
MainActivity.smali 파일을 수정 한 뒤, Recompile 한 App이 공격자가 의도한 동작(문자열 박스 출력)을 수행하는 것을 확인할 수 있다.



-본 실습에서는 수정한 App이 위협적인 동작을 수행하진 않지만, 권한이 없는 데이터에 접근하거나, 특정 Target에 공격을 수행하도록 App을 수정하면 치명적인 동작을 수행할 수 있다.

5. 취약점 대응 방안 설명

Application Pathing에 의한 취약점에 의해 Application이 위조 또는 변조 되는 것에 대응하는 방법 중 하나는 소스코드를 난독화 하는 것이다.
Android Studio에서 기본으로 제공하는 도구인 프로가드와 같은 난독화 도구를 사용하여, Application을 난독화 한 이후 Packaging하여 배포하여 취약점을 방어할 수 있다.

<어플리케이션 #2>

1. 타겟 어플리케이션 설명

-DIVA(Damn Insecure And Vulnerable App)은 의도적으로 안전하지 않도록 설계된 Application이다.

-해당 App은 Application을 설계하는 과정에서 안전하지 않은 습관들이 어떤 취약점을 발생시키는지 알 수 있도록 여러가지 보안 취약점으로 구성되어 있다.

-본 실습에서는 Insecure Logging 취약점에 대한 공격을 재현하고, 취약점을 분석할 예정이다.

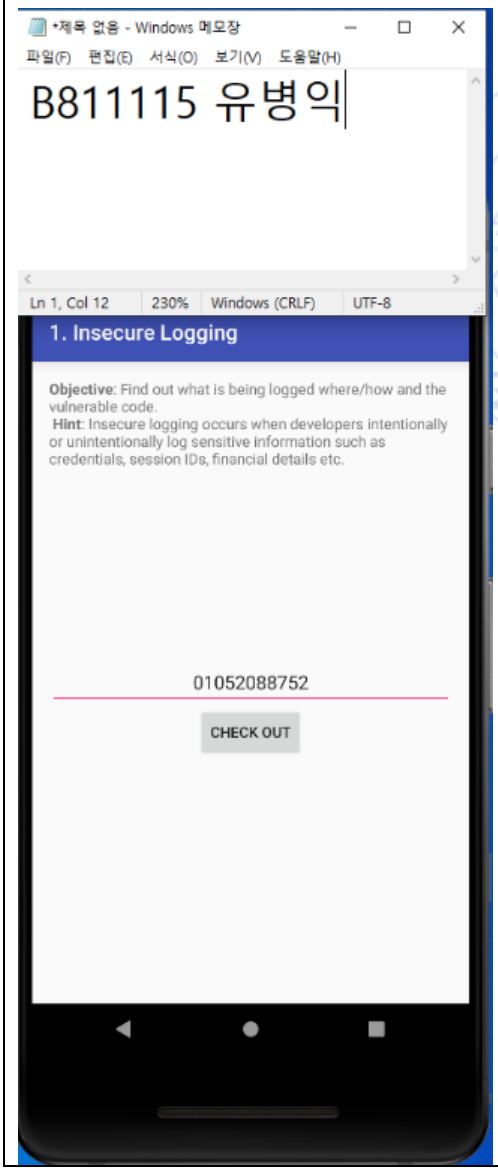
2. 취약점에 대한 공격 재현

#Application Install

-https://github.com/0xArab/diva-apk-file 에서 제공하는 DivaApplication.apk 를 사용해 AVD 환경에 App을 install 한다.

#Card Number

-임의의 카드 번호를 입력하고 "CHECK OUT" 버튼을 클릭하면 Error Message가 발생한다.

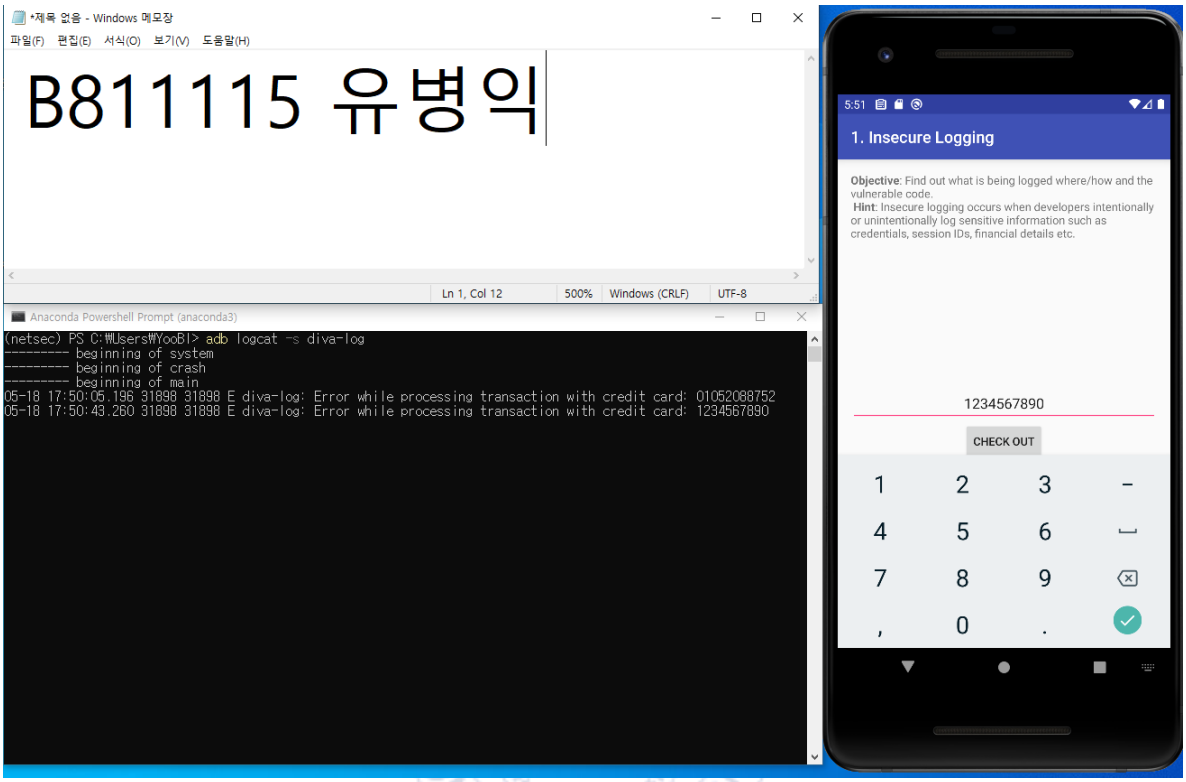


#Log

-adb logcat -s diva-log 명령어를 실행하여 로그를 확인한다.

-다른 번호를 입력하고 "adb logcat -s diva-log" 명령어를 실행해본다.

-입력한 카드번호가 평문 그대로 Log로 출력되는 것을 확인할 수 있다.



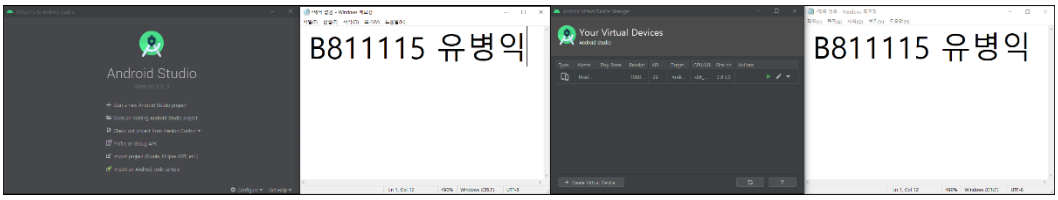
3. 취약점 분석에 사용한 모든 도구 및 분석 방법 설명

#Android Studio

-Android Application 개발을 위해 Jetbrains의 IntelliJ를 기반으로 만든

-AVD 가상 환경에서 App을 설치하고 구동하는 과정을 위해 사용한다.

-Android Studion를 통해 PC환경에서 Application의 구동하고, 특정 작업을 수행하도록 하여 그 과정을 관찰할 수 있다.



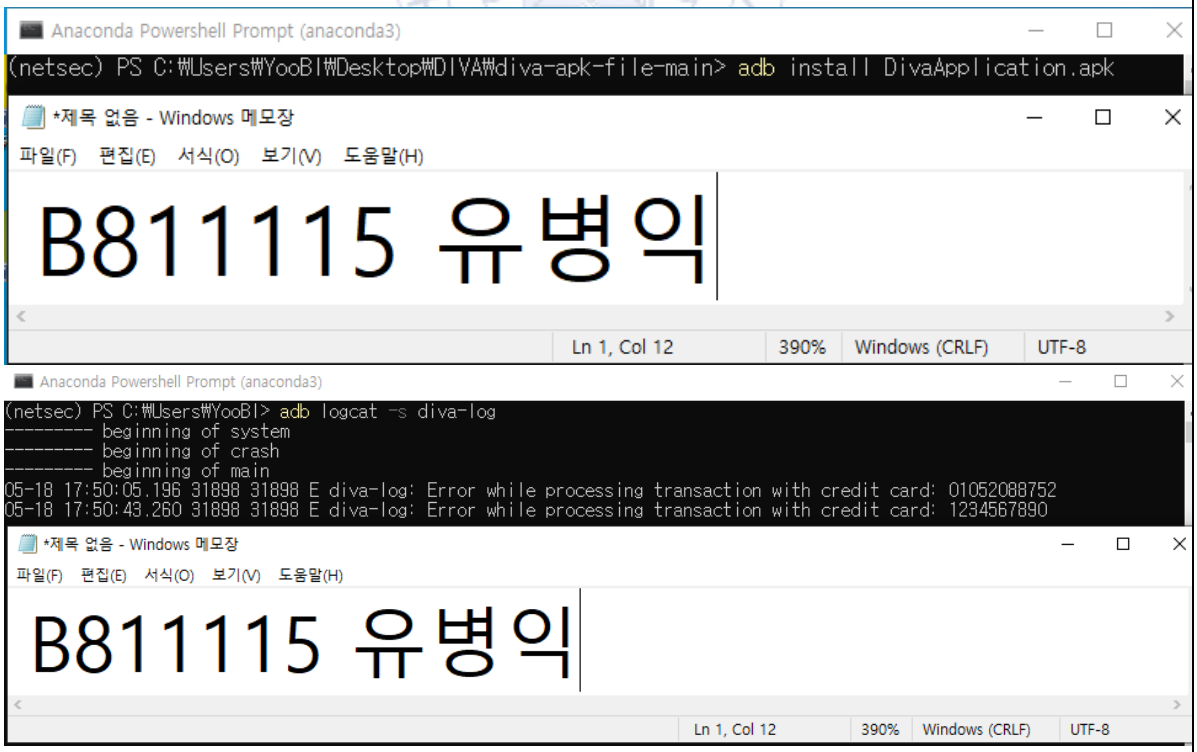
#APK Easy Tool

-Apk Easy Tool은 Application의 APK 파일을 관리, 서명, 컴파일 및 디컴파일할 수 있는 GUI Application이다.

-해당 실습에서 Apk Easy Tool을 활용해 Apk파일을 Decompile 하고 수정한 파일을 Recompile 할 때 사용한다

#ADB

- 안드로이드 단말기와 데스크톱 간에 통신을 할 때 필요한 도구.
- adb 명령어를 사용하여 App 설치 및 제거, Log 확인 등의 작업에 사용한다.

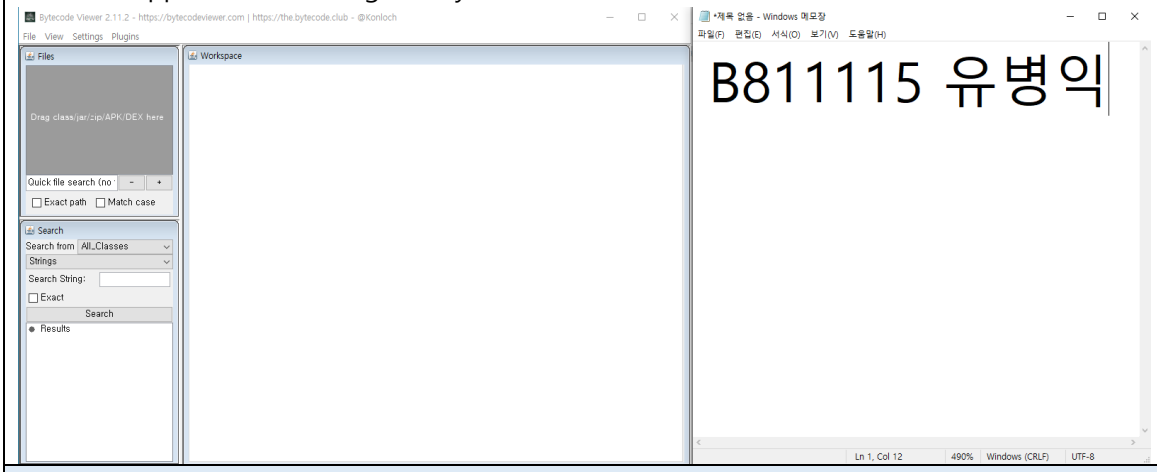


#ByteCodeViewer

-Java, Android 등 소스코드 리버싱 분석 도구이다.

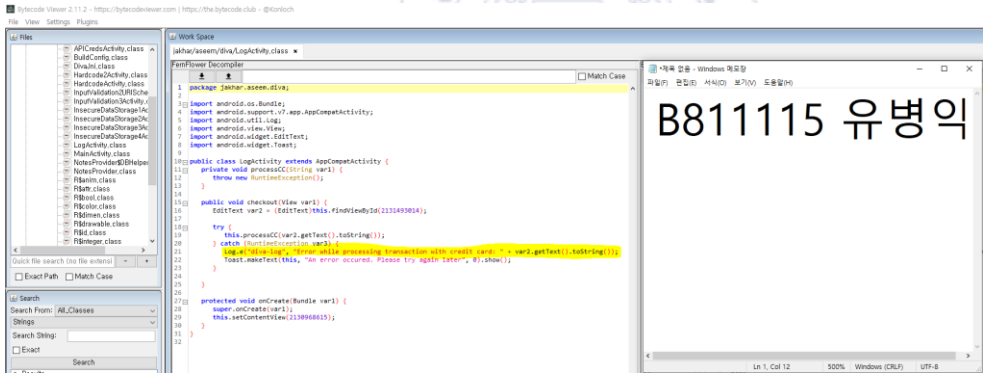
-APK파일을 Decompile하여 바이트코드 형태와 클래스 파일을 자바 파일로 변환하여 소스파일로 복원하여 같이 보여준다.

-DIVA Application의 LogActivity 클래스의 코드를 확인하는 과정에서 사용한다.

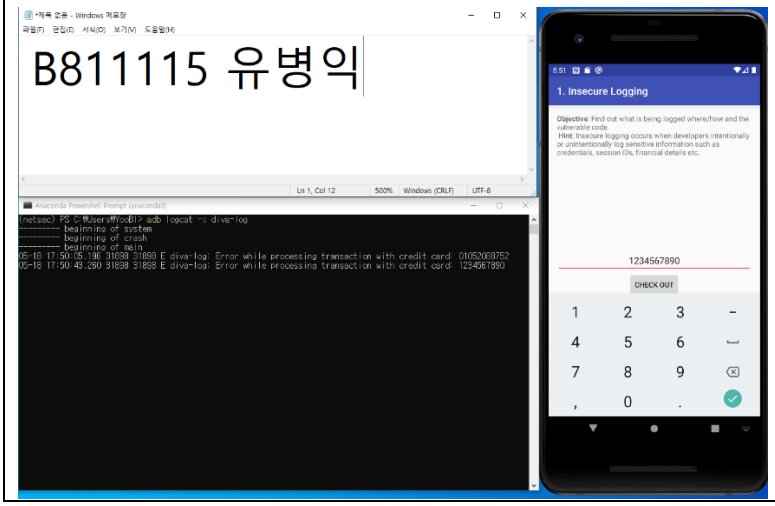


4. 취약점 분석 결과 설명

-LogActivity 클래스의 코드를 확인해보면, 사용자가 입력한 카드 번호에 대한 정보가 평문 그대로 로그로 출력하도록 하는 코드를 확인할 수 있다. (Log.e 함수 부분)



-로그에 중요한 정보들이 평문 그대로 남게 되면, 공격자에 의해 중요한 정보가 유출될 위험이 존재한다.



5. 취약점 대응 방안 설명
-Log와 관련있는 코드를 작성할 때, 중요한 정보라고 판단되는 것들은 Logging 하지 않는다. -중요한 정보들이 Log에 남는 것을 확인하게 된다면, 해당 정보를 Logging하는 부분의 소스코드를 찾아, 해당 내용이 출력되지 않도록 수정한다.(Log부분을 제거한다.)