

<네트워크보안 실습 과제 (2) - B>

학번 / 학과: B911012 / 컴퓨터공학과

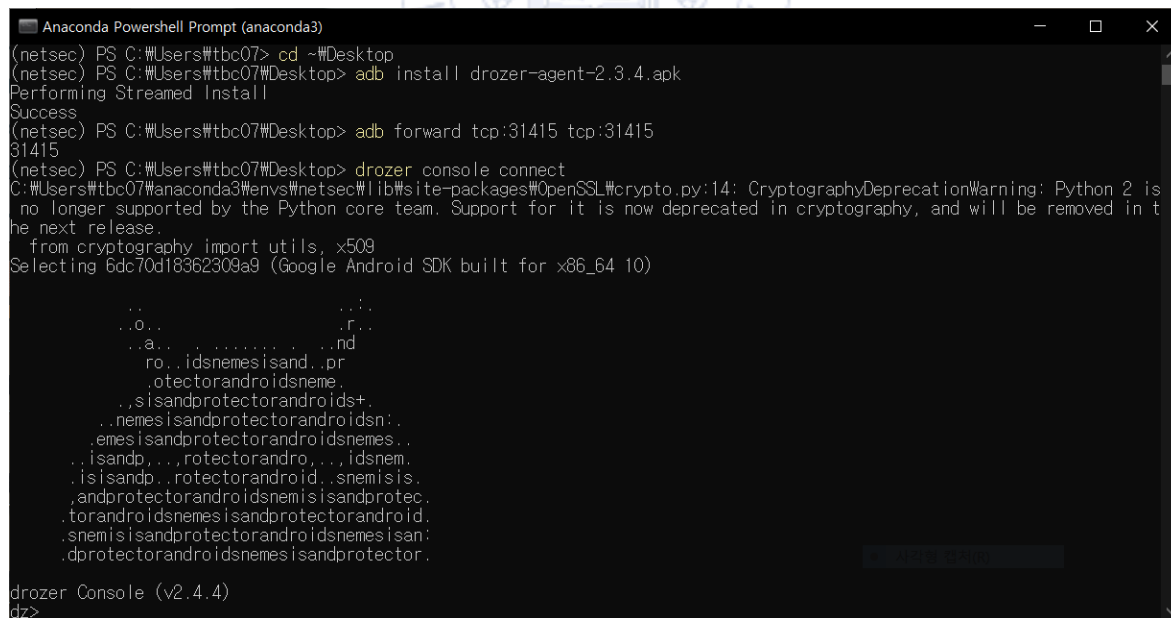
이름: 금예인

1. 구현 어플리케이션 설명

간단한 일기장 어플리케이션을 구현하였습니다. 특정 날짜를 선택하여 일기를 쓸 수 있으며, 하루의 기분을 Rating 방식으로 기록하여 추후 함께 조회할 수 있도록 만들었습니다. Splash screen이 끝나면 캘린더가 있는 메인 화면이 등장하고, 캘린더에서 날짜를 선택하면 해당 날짜 일기를 쓸 수 있는 버튼이 나타납니다. 하단 네비게이션 바 왼쪽의 Diary를 선택하여 작성했던 일기를 볼 수 있습니다. SQLite를 활용해 날짜, 제목, 내용 등을 입력하고 일기를 저장할 수 있도록 구현하였습니다. 또한 하단 네비게이션 바 오른쪽의 Setting을 선택하여 어플리케이션 비밀번호를 설정할 수 있습니다. 네 자리 숫자 패스워드를 설정하고 변경하거나, 이를 해제할 수 있습니다.

2. 취약점에 대한 공격 재현

실습과제1에서 사용했던 Drozer를 활용하였습니다.



```
Anaconda Powershell Prompt (anaconda3)
(netsec) PS C:\Users\tbc07> cd ~\Desktop
(netsec) PS C:\Users\tbc07\Desktop> adb install drozer-agent-2.3.4.apk
Performing Streamed Install
Success
(netsec) PS C:\Users\tbc07\Desktop> adb forward tcp:31415 tcp:31415
31415
(netsec) PS C:\Users\tbc07\Desktop> drozer console connect
C:\Users\tbc07\anaconda3\envs\netsec\Lib\site-packages\OpenSSL\crypto.py:14: CryptographyDeprecationWarning: Python 2 is
no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in t
he next release.
  from cryptography import utils, x509
Selecting 6dc70d18362309a9 (Google Android SDK built for x86_64 10)

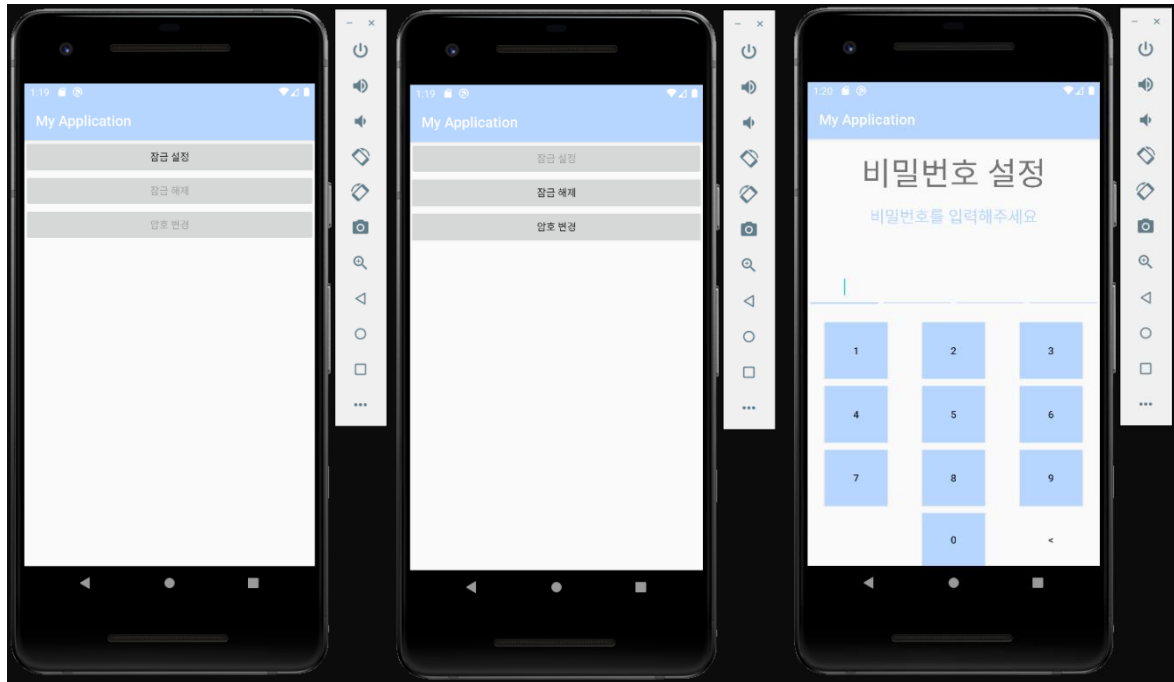
..
..0..
..a..
..a..
..ro..idsnemesisand..pr
..rotectorandroidsneme
..sisandprotectorandroids+
..nemesisandprotectorandroidsn:
..emesisandprotectorandroidsnemes..
..isandp,..rotectorandro,..idsnem.
..isisandp,..rotectorandroid,..snemis.
..andprotectorandroidsnemisandprotec.
..torandroidsnemesisandprotectorandroid.
..snemisandprotectorandroidsnemisand:
..dprotectorandroidsnemesisandprotector.

drozer Console (v2.4.4)
dz>
```

1. Drozer 설치 및 실행

```
Anaconda Powershell Prompt (anaconda3)
(base) PS C:\Users\wtbc07> conda activate netsec
(netsec) PS C:\Users\wtbc07> cd ~\Desktop
(netsec) PS C:\Users\wtbc07\Desktop> adb install app-release.apk
Performing Streamed Install
Success
```

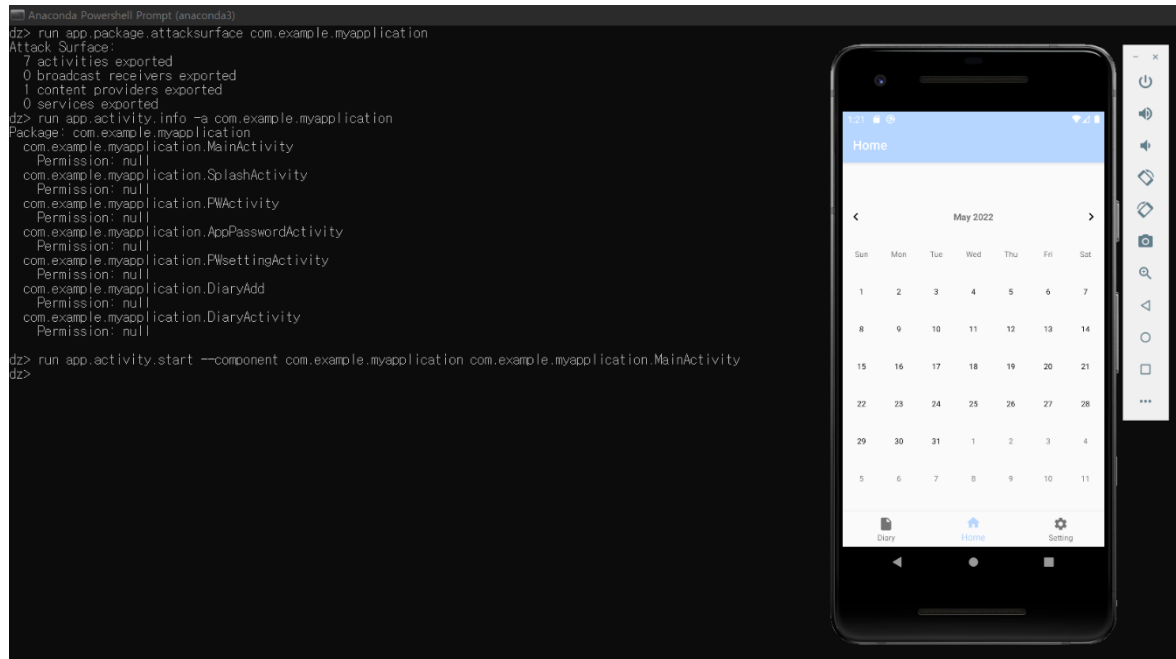
2. myapplication 설치



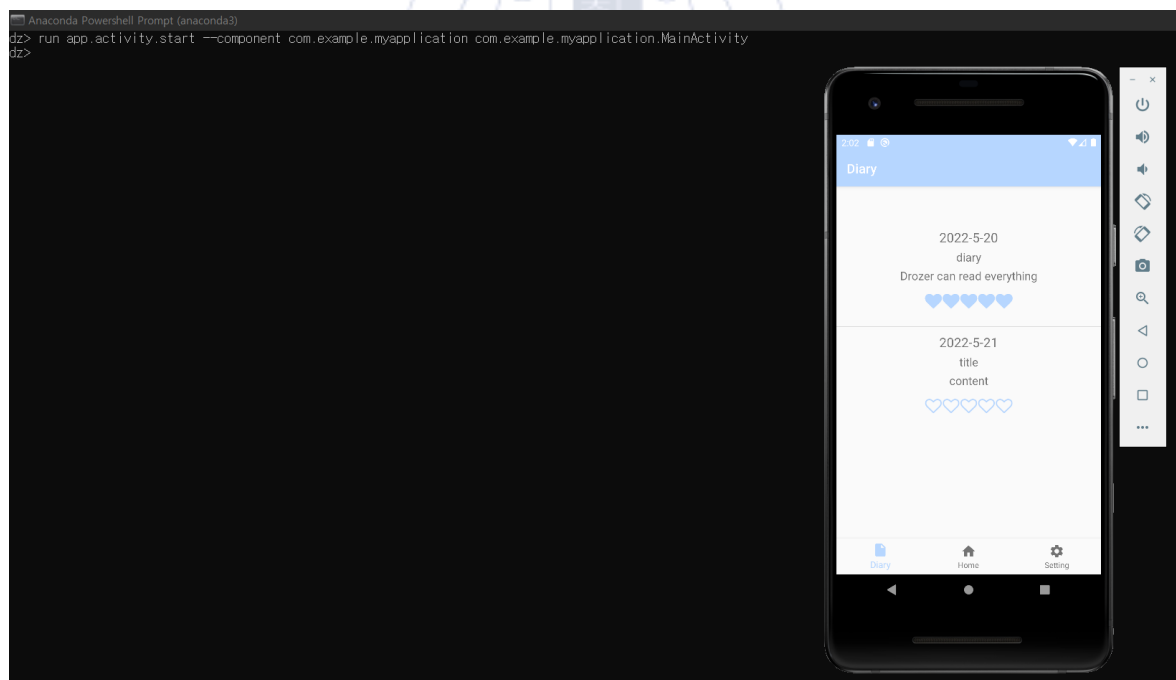
3. myapplication 비밀번호 설정 완료

```
Anaconda Powershell Prompt (anaconda3)
dz> run app.package.attacksurface com.example.myapplication
Attack Surface:
  7 activities exported
  0 broadcast receivers exported
  1 content providers exported
  0 services exported
dz> run app.activity.info -a com.example.myapplication
Package: com.example.myapplication
com.example.myapplication.MainActivity
  Permission: null
com.example.myapplication.SplashActivity
  Permission: null
com.example.myapplication.PWActivity
  Permission: null
com.example.myapplication.AppPasswordActivity
  Permission: null
com.example.myapplication.PWSettingActivity
  Permission: null
com.example.myapplication.DiaryAdd
  Permission: null
com.example.myapplication.DiaryActivity
  Permission: null
dz>
```

4. Drozer를 통한 myapplication의 노출된 컴포넌트 검색



5. Drozer를 통한 MainActivity 실행



6. Drozer를 통한 일기 조회

3. 구현 어플리케이션 취약점 설명

'2. 취약점에 대한 공격 재현'에서 myapplication은 어플리케이션 비밀번호를 설정해둔 상태였습니다. 따라서 어플 실행 시 설정해둔 비밀번호를 입력해야, 다음 화면으로 전환될 수 있었습니다.

```

Anaconda Powershell Prompt (anaconda3)
dz> run app.package.attacksurface com.example.myapplication
Attack Surface:
  7 activities exported
  0 broadcast receivers exported
  1 content providers exported
  0 services exported
dz> run app.activity.info -a com.example.myapplication
Package: com.example.myapplication
com.example.myapplication.MainActivity
  Permission: null
com.example.myapplication.SplashActivity
  Permission: null
com.example.myapplication.PWActivity
  Permission: null
com.example.myapplication.AppPasswordActivity
  Permission: null
com.example.myapplication.PWsettingActivity
  Permission: null
com.example.myapplication.DiaryAdd
  Permission: null
com.example.myapplication.DiaryActivity
  Permission: null
dz>

```

하지만 Drozer를 통해 노출된 컴포넌트 정보를 조회하였을 때 일기 등록, 열람, 그리고 비밀번호 설정이 가능한 MainActivity가 노출되어 있었고, 모든 권한이 null인 것을 확인할 수 있었습니다.

```

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="My Application"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <provider
        android:authorities="com.example.myapplication"
        android:name=".DiaryProvider"
        android:exported="true"
        android:readPermission="null"
        android:writePermission="null"/>

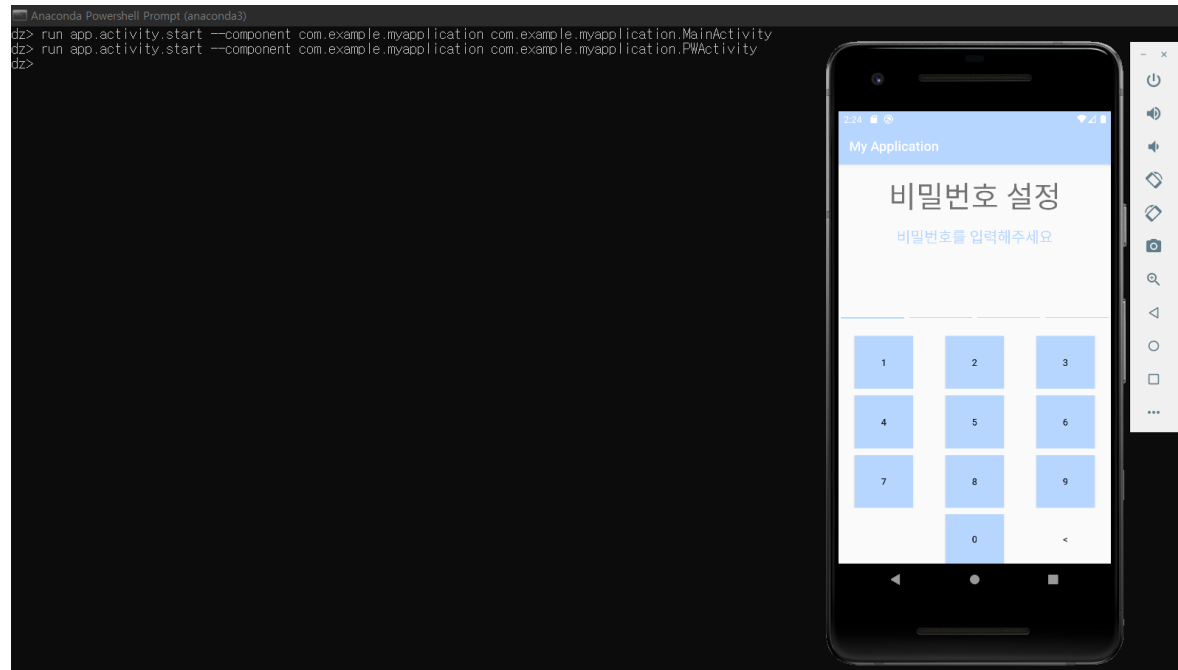
    <activity android:name=".MainActivity" android:exported="true">

```

실제 Manifest.xml 파일에서도, MainActivity의 android:exported 속성이 "true"로 설정되어 있는 것을 확인할 수 있습니다.

따라서 어플리케이션의 정상적인 인증 절차를 우회하여 비밀번호를 입력하지 않아도 접근 권한을 획득할 수 있었습니다. 즉, 다른 앱에서.myapplication의 특정 Activity에 접근할 수 있고,.myapplication은 외부에서 발생하는 intent에도 영향을 받을 수 있습니다.

이러한 액티비티 컴포넌트의 취약점과 취약한 인증 매커니즘으로 인해 공격자는 사용자가 작성하지 않은 글을 등록할 수 있고, 사용자의 글을 모두 조회할 수 있습니다. 만약 비밀번호가 설정되어 있지 않은 상태라면, 다음 screenshot처럼 사용자가 어플리케이션을 정상적으로 이용할 수 없도록 비밀번호를 설정할 수도 있습니다.



activity의 android:exported 속성을 'false'로 지정하여 이러한 취약점을 방어할 수 있습니다.