CyberHost internship final project report

Name: George David D

1. **Vulnerability Name** : Insecure permission and flags set in the Android manifest file

Vulnerable App : AndroGoat

Severity Level **:** Medium

Summary: The Android Manifest file has app permission, Flags, and component API-related information. <uses-permission **android:name="android.permission.WRITE_EXTERNAL_STORAGE"/> and <application android:allowBackup="true" android:debuggable="true"** this

write external storage permission will give the permission to save or import permission to the app and if allows taking a backup of the data insecure way and debug contains read, write, and execute permission-related information, both flags should be set as **false.**

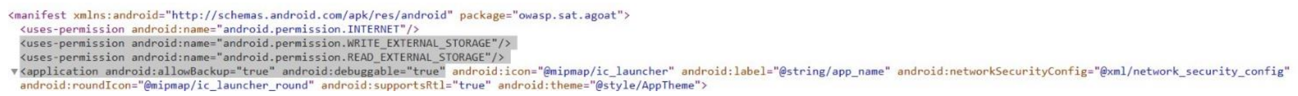
Steps to reproduce

1. Decompile the application using APKTOOL with Appie framework
2. Check for Android permission and allow backup and debug flags
3. If it is true the app is vulnerable
4. adb backup -f report.ab -appie use the APPIe command to pull the backup files
5. Using an Android backup extractor to get the database files
6. Java -jar abe.jar unpack test.ab test.tar

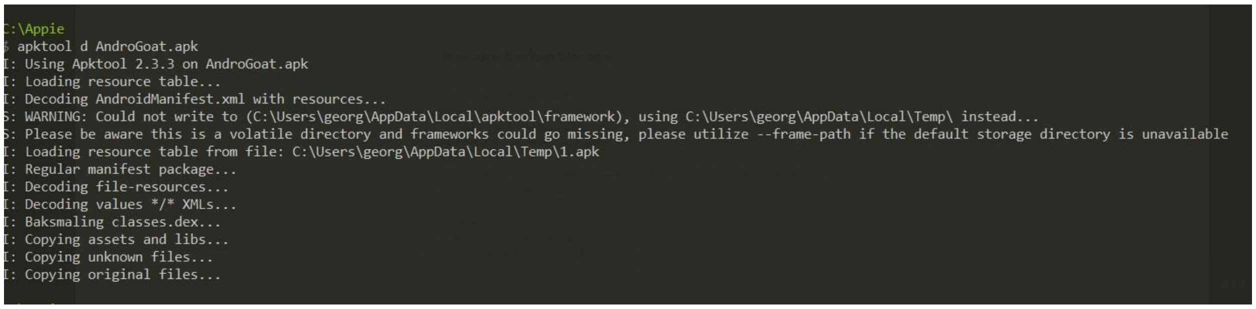
7. Extract the test rar to get database files.
8. To exploit debug=" true" install arm translator
9. run-as owasp.sat.agot
10.

Poc

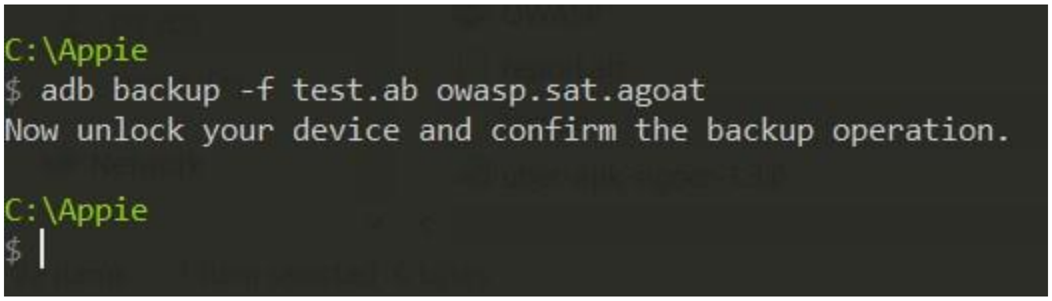
insecure backup



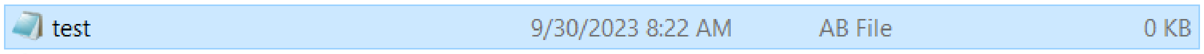
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="owasp.sat.agoat">
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
▼<application android:allowBackup="true" android:debuggable="true" android:icon="@mipmap/ic_launcher" android:label="@string/app_name" android:networkSecurityConfig="@xml/network_security_config"
  android:roundIcon="@mipmap/ic_launcher_round" android:supportsRtl="true" android:theme="@style/AppTheme">
```



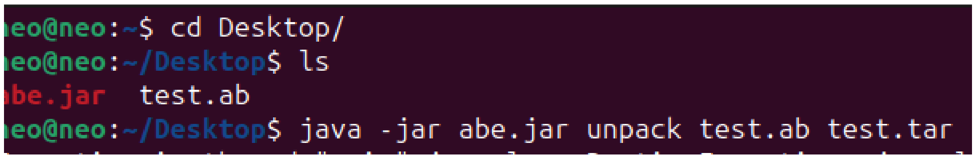
```
C:\Appie
$ apktool d AndroGoat.apk
I: Using Apktool 2.3.3 on AndroGoat.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
S: WARNING: Could not write to (C:\Users\georg\AppData\Local\apktool\framework), using C:\Users\georg\AppData\Local\Temp\ instead...
S: Please be aware this is a volatile directory and frameworks could go missing, please utilize --frame-path if the default storage directory is unavailable
I: Loading resource table from file: C:\Users\georg\AppData\Local\Temp\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```



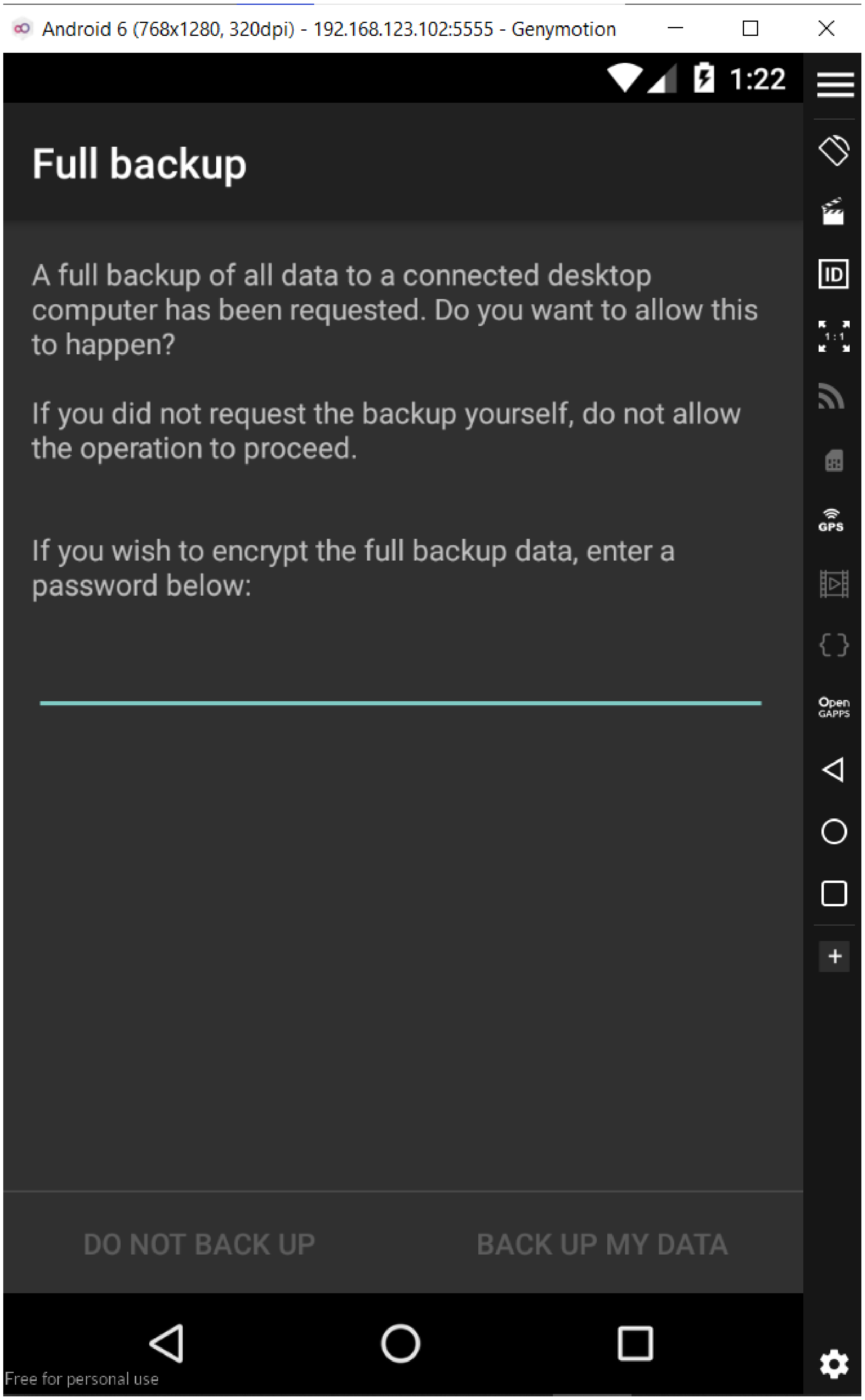
```
C:\Appie
$ adb backup -f test.ab owasp.sat.agoat
Now unlock your device and confirm the backup operation.

C:\Appie
$ |
```



```
test                                9/30/2023 8:22 AM        AB File                0 KB
```



```
eo@neo:~$ cd Desktop/
eo@neo:~/Desktop$ ls
be.jar   test.ab
eo@neo:~/Desktop$ java -jar abe.jar unpack test.ab test.tar
```


Android 6 (768x1280, 320dpi) - 192.168.123.102:5555 - Genymotion

1:22

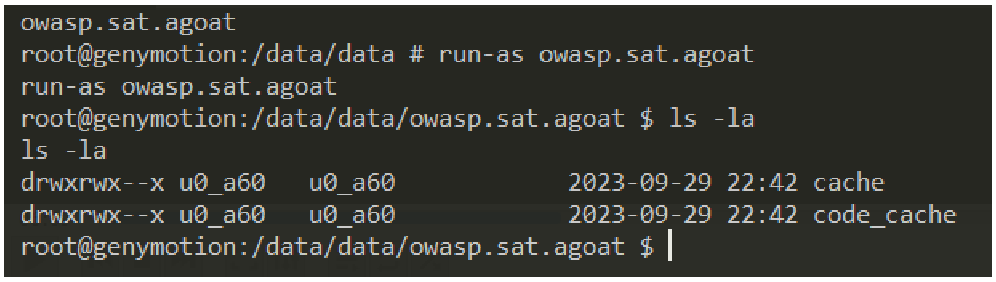
Full backup

A full backup of all data to a connected desktop computer has been requested. Do you want to allow this to happen?

If you did not request the backup yourself, do not allow the operation to proceed.

If you wish to encrypt the full backup data, enter a password below:

2. **debug detection**



```
owasp.sat.agoat
root@genymotion:/data/data # run-as owasp.sat.agoat
run-as owasp.sat.agoat
root@genymotion:/data/data/owasp.sat.agoat $ ls -la
ls -la
drwxrwx--x u0_a60   u0_a60           2023-09-29 22:42 cache
drwxrwx--x u0_a60   u0_a60           2023-09-29 22:42 code_cache
root@genymotion:/data/data/owasp.sat.agoat $
```

Impact

Backup= true it will lead to the attacker taking user application backup data and database files

Recommendation

In the manifest file set flags false for debug and backup

3. **Vulnerability name: No Code obfuscation**

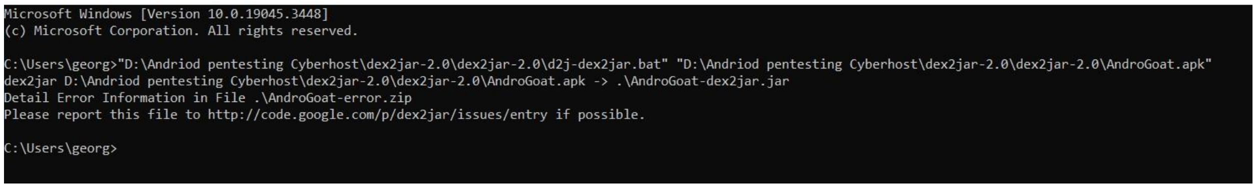
Summary:

Code obfuscation is the process of encrypting and complicating lines of code, data, and communication loops. These measures cause hackers immense difficulty in interpreting and changing existing information. Ultimately, obfuscation stymies potential hackers, limiting their access and ability to steal and manipulate.

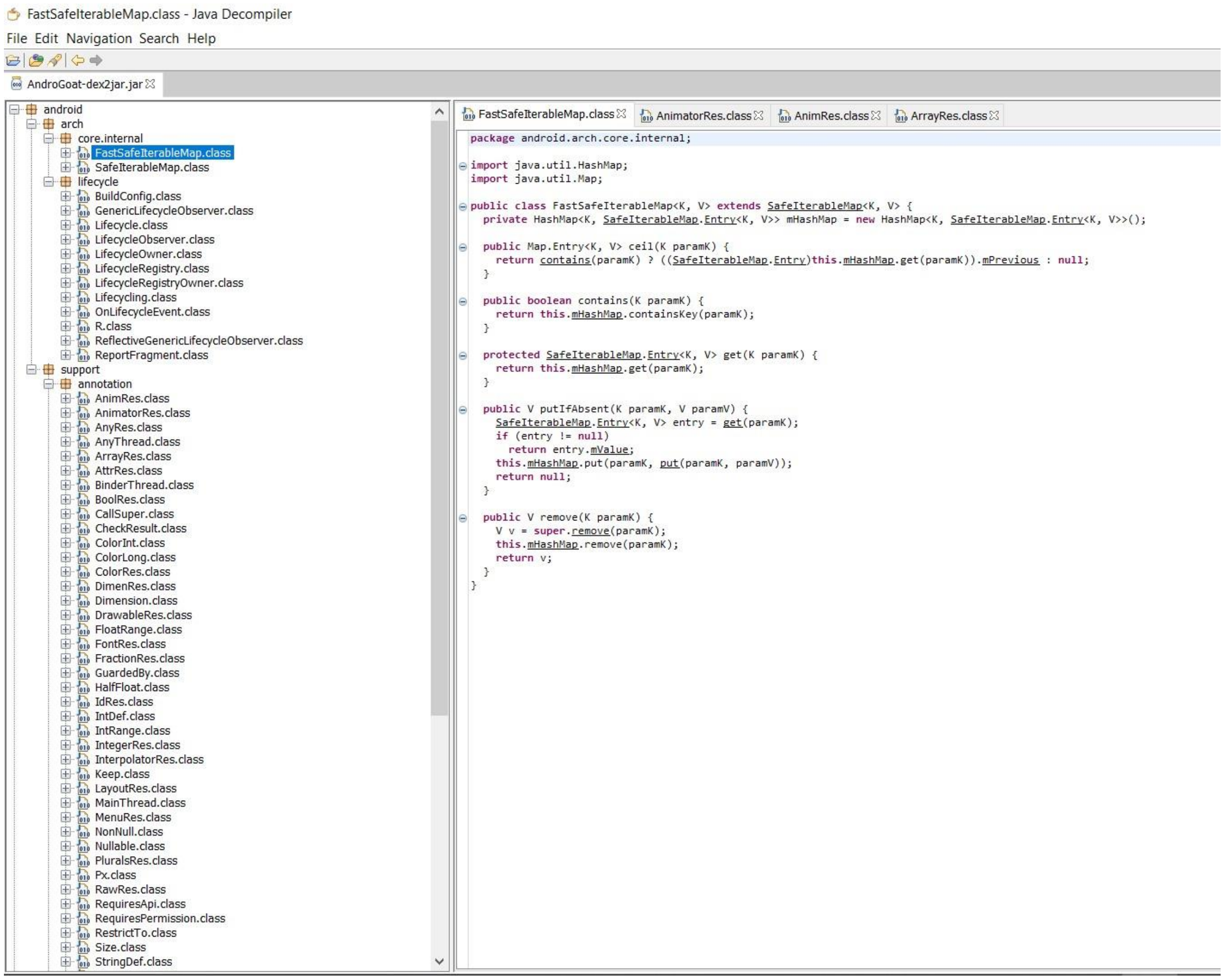
Steps to reproduce

1. Using dex2jar to decompile APK build to source code
2. Navigate the dex2jar folder in CMD run with the target application
3. It will return the jar file of the app
4. Open jar in jd-gui

POC



```
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\georg>"D:\Andriod pentesting Cyberhost\dex2jar-2.0\dex2jar-2.0\d2j-dex2jar.bat" "D:\Andriod pentesting Cyberhost\dex2jar-2.0\dex2jar-2.0\AndroGoat.apk"
dex2jar D:\Andriod pentesting Cyberhost\dex2jar-2.0\dex2jar-2.0\AndroGoat.apk -> .\AndroGoat-dex2jar.jar
Detail Error Information in File .\AndroGoat-error.zip
Please report this file to http://code.google.com/p/dex2jar/issues/entry if possible.

C:\Users\georg>
```


FastSafeIterableMap.class - Java Decompiler
File Edit Navigation Search Help
AndroGoat-dex2jar.jar
FastSafeIterableMap.class
AnimatorRes.class
AnimRes.class
ArrayRes.class

```
android
  arch
    core.internal
      FastSafeIterableMap.class
      SafeIterableMap.class
    lifecycle
      BuildConfig.class
      GenericLifecycleObserver.class
      Lifecycle.class
      LifecycleObserver.class
      LifecycleOwner.class
      LifecycleRegistry.class
      LifecycleRegistryOwner.class
      Lifecycling.class
      OnLifecycleEvent.class
      R.class
      ReflectiveGenericLifecycleObserver.class
      ReportFragment.class
  support
    annotation
      AnimRes.class
      AnimatorRes.class
      AnyRes.class
      AnyThread.class
      ArrayRes.class
      AttrRes.class
      BinderThread.class
      BoolRes.class
      CallSuper.class
      CheckResult.class
      ColorInt.class
      ColorLong.class
      ColorRes.class
      DimenRes.class
      Dimension.class
      DrawableRes.class
      FloatRange.class
      FontRes.class
      FractionRes.class
      GuardedBy.class
      HalfFloat.class
      IdRes.class
      IntDef.class
      IntRange.class
      IntegerRes.class
      InterpolatorRes.class
      Keep.class
      LayoutRes.class
      MainThread.class
      MenuRes.class
      NonNull.class
      Nullable.class
      PluralsRes.class
      Px.class
      RawRes.class
      RequiresApi.class
      RequiresPermission.class
      RestrictTo.class
      Size.class
      StringDef.class
```

```java
package android.arch.core.internal;

import java.util.HashMap;
import java.util.Map;

public class FastSafeIterableMap<K, V> extends SafeIterableMap<K, V> {
  private HashMap<K, SafeIterableMap.Entry<K, V>> mHashMap = new HashMap<K, SafeIterableMap.Entry<K, V>>();

  public Map.Entry<K, V> ceil(K paramK) {
    return contains(paramK) ? ((SafeIterableMap.Entry)this.mHashMap.get(paramK)).mPrevious : null;
  }

  public boolean contains(K paramK) {
    return this.mHashMap.containsKey(paramK);
  }

  protected SafeIterableMap.Entry<K, V> get(K paramK) {
    return this.mHashMap.get(paramK);
  }

  public V putIfAbsent(K paramK, V paramV) {
    SafeIterableMap.Entry<K, V> entry = get(paramK);
    if (entry != null)
      return entry.mValue;
    this.mHashMap.put(paramK, put(paramK, paramV));
    return null;
  }

  public V remove(K paramK) {
    V v = super.remove(paramK);
    this.mHashMap.remove(paramK);
    return v;
  }
}
```

Impact

The attacker will able to get the source code of the application and make MOD or Reverse engineering and unethical purpose

Recommendation

Implement code obfuscation

4. Vulnerability Name: HTML injection

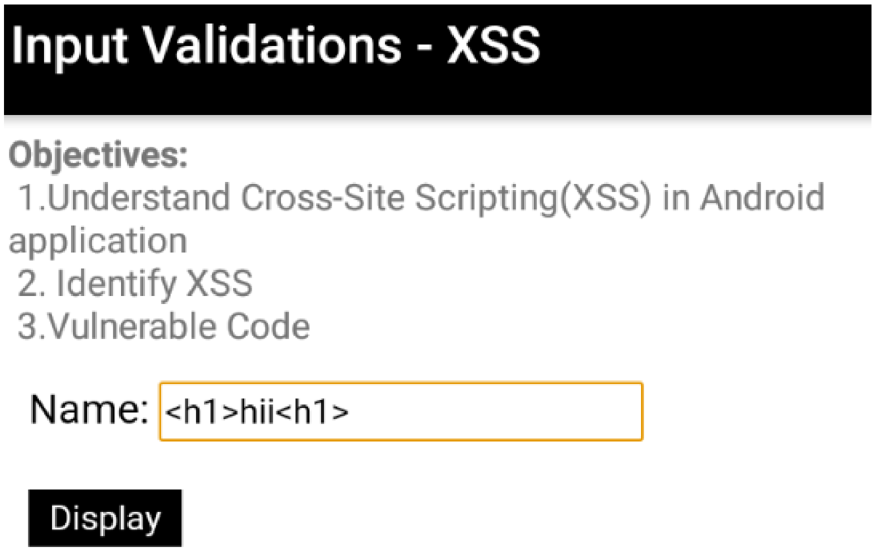
Summary:

HTML injection, also known as cross-site scripting (XSS), is a common vulnerability that occurs when user-generated input is not properly handled or sanitized.

Steps to reproduce:

1. Enter injection payload to any of the input field in an application

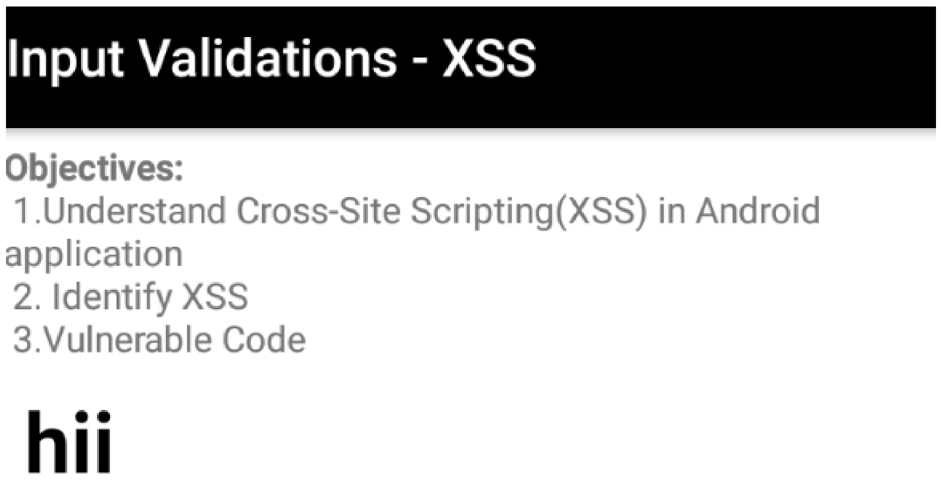
POC



Input Validations - XSS

Objectives:
1.Understand Cross-Site Scripting(XSS) in Android application
2. Identify XSS
3.Vulnerable Code

Name: <h1>hii<h1>


Input Validations - XSS

Objectives:
1.Understand Cross-Site Scripting(XSS) in Android application
2. Identify XSS
3.Vulnerable Code

hii

Impact

Injection it will lead to session cookie hijacking credentials theft and login bypass (SQL injection)

Recommendation

Use proper validation and firewall to block injection payloads