JOEL FRANKO

7338828097

joelfranko1234@gmail.com

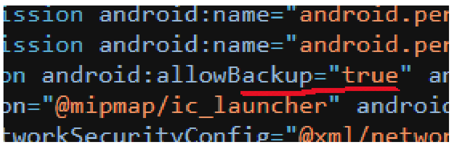
Vulnerability Analysis

Overview:

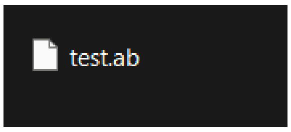
I did a Vulnerability Analysis and Penetration testing on android app Andro Goat and found some vulnerabilities and mentioned those vulnerabilities and its effects to the app in this report.

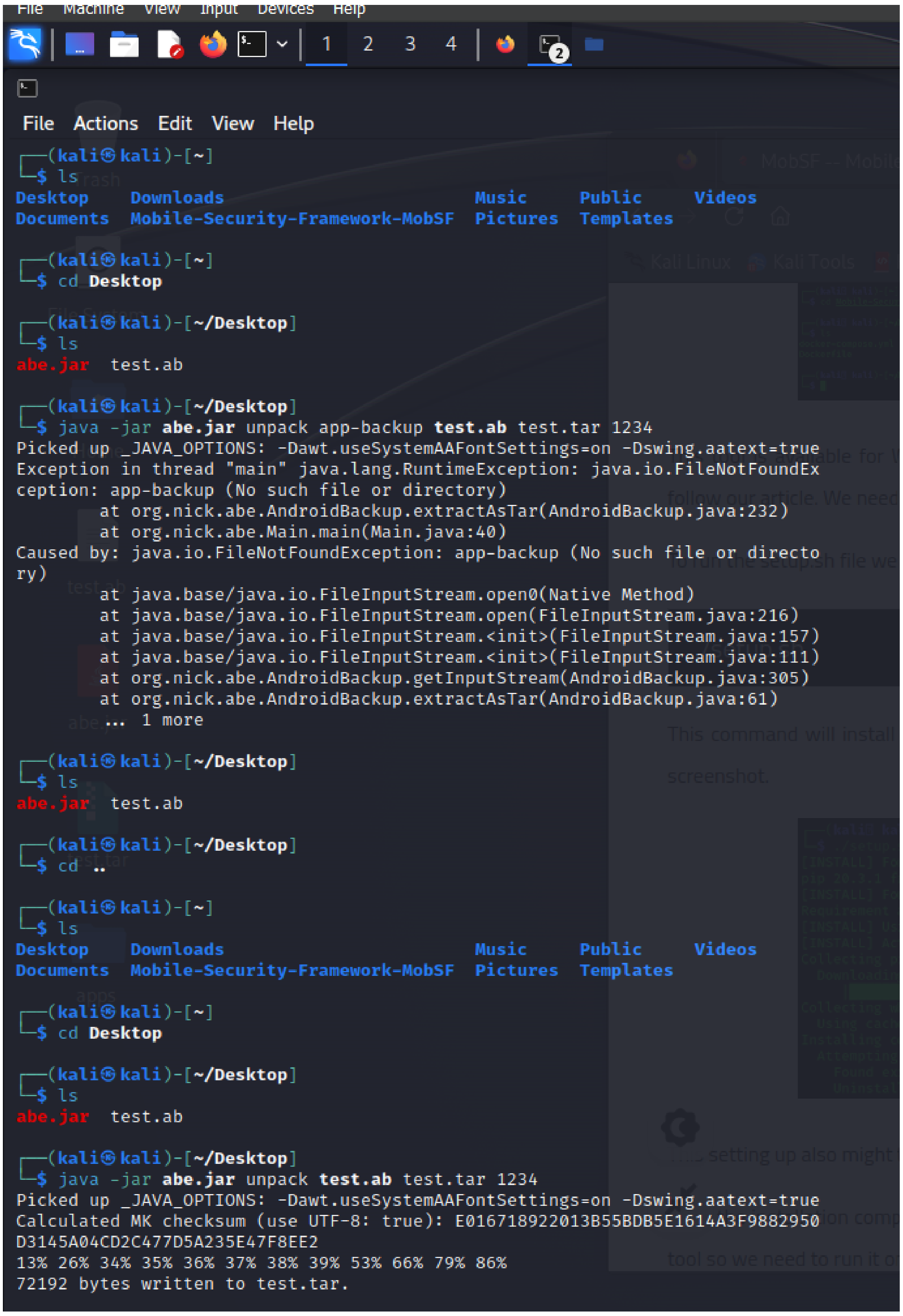
1. Insecure Backup Storage:
    - Vulnerable Application – Andro Goat.
    - Vulnerability Description:

      Insecure data storage vulnerabilities occur when development teams assume that users or malware will not have access to a mobile device's filesystem and subsequent sensitive information in data-stores on the device and let the backup to be open.
    - Steps to Reproduce:
        - Step 1: Open Appie and extract the source code of the app.
        - Step 2: Open AndroidManifest.xml file and found that backup is allowed.
        - Step 3: Now get the backup file through Appie using "adb backup -f test.ab AndroGoat" command
        - Step 4: Then use Kali to extract the complete application.
    - Impact:
        - Attackers can get the complete information about the application.
        - Attackers can use it to exploit the app
    - Proof of Concept:



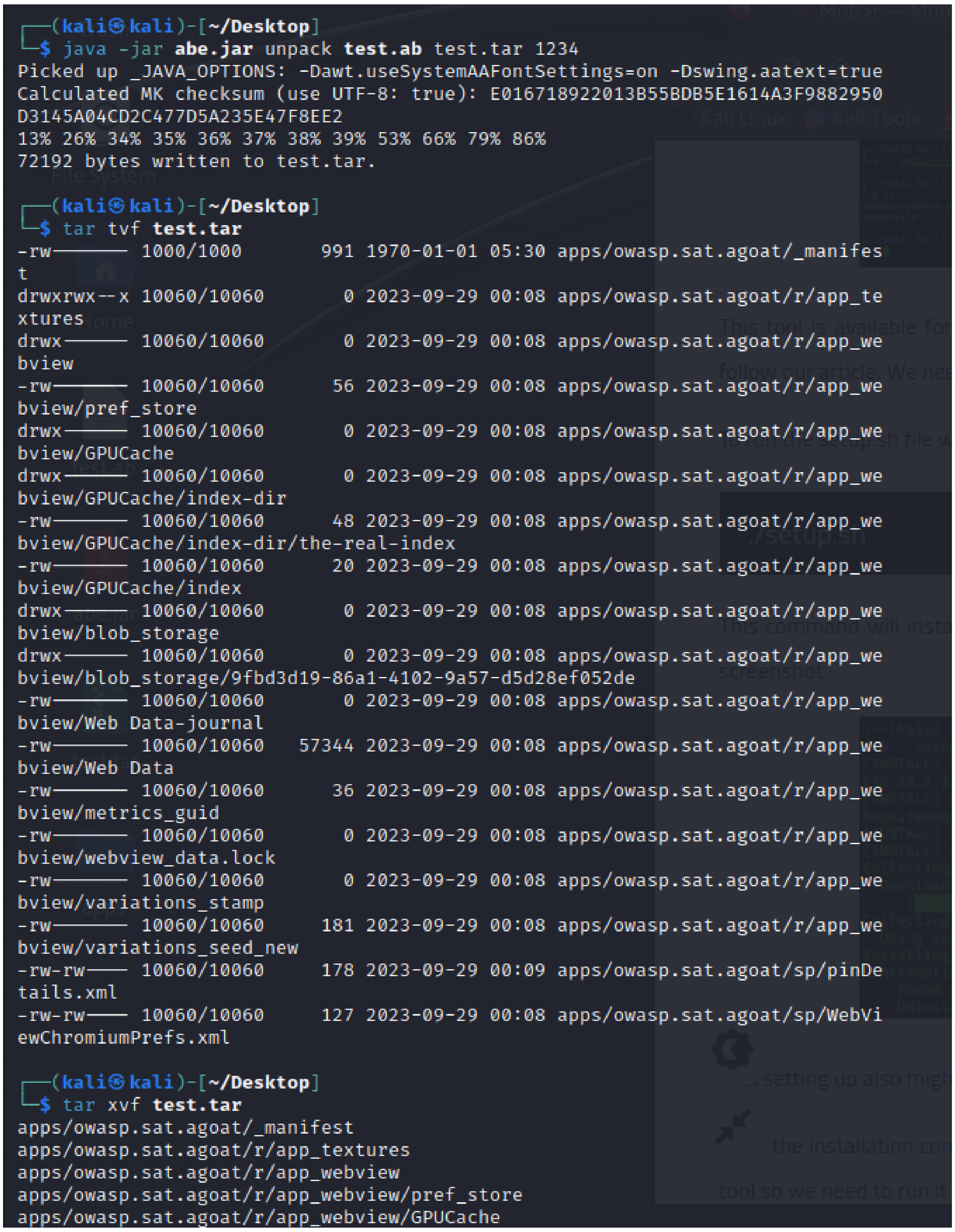
```
ission android:name="android.per
ission android:name="android.per
on android:allowBackup="true" ar
on="@mipmap/ic_launcher" androic
tworkSecurityConfig="@xml/networ
```



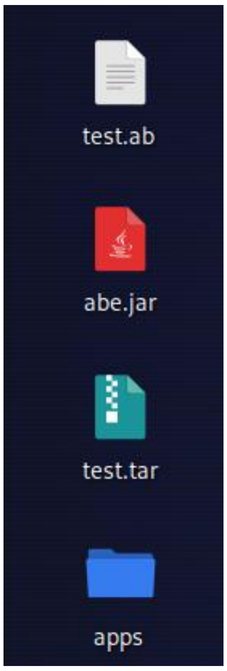


File Machine View Input Devices Help
1 2 3 4
File Actions Edit View Help

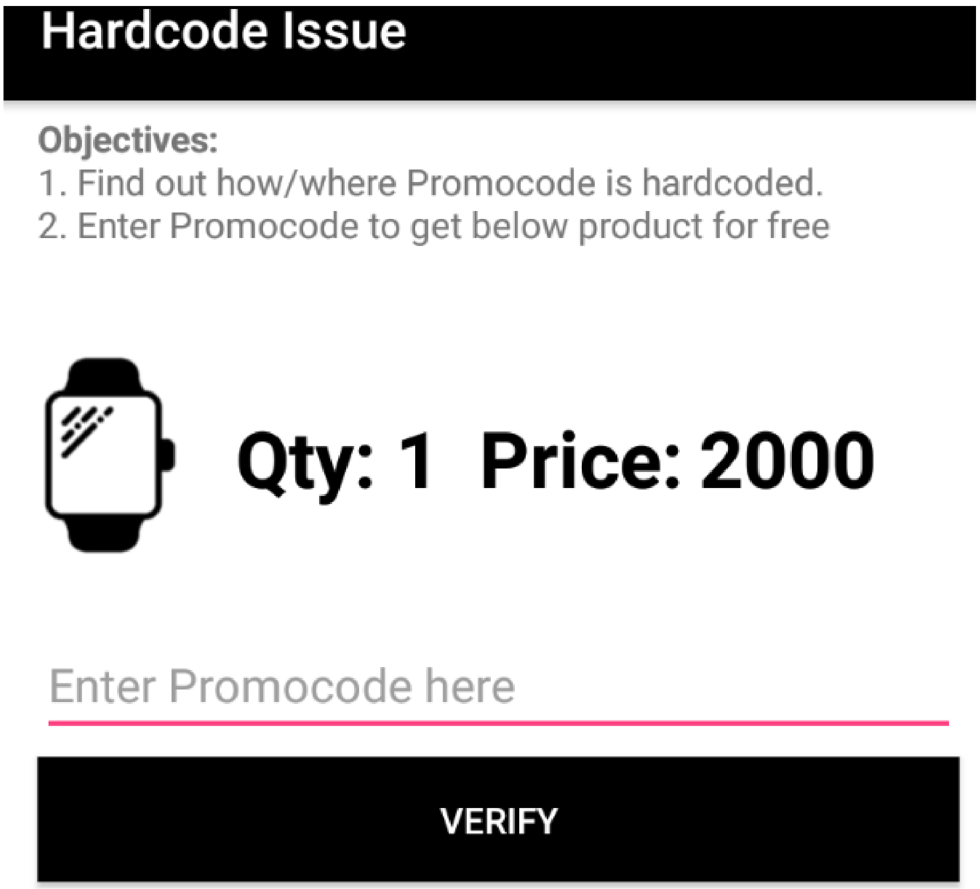
```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads                        Music    Public     Videos
Documents  Mobile-Security-Framework-MobSF  Pictures Templates

┌──(kali㉿kali)-[~]
└─$ cd Desktop

┌──(kali㉿kali)-[~/Desktop]
└─$ ls
abe.jar  test.ab

┌──(kali㉿kali)-[~/Desktop]
└─$ java -jar abe.jar unpack app-backup test.ab test.tar 1234
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Exception in thread "main" java.lang.RuntimeException: java.io.FileNotFoundEx
ception: app-backup (No such file or directory)
        at org.nick.abe.AndroidBackup.extractAsTar(AndroidBackup.java:232)
        at org.nick.abe.Main.main(Main.java:40)
Caused by: java.io.FileNotFoundException: app-backup (No such file or directo
ry)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:216)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:111)
        at org.nick.abe.AndroidBackup.getInputStream(AndroidBackup.java:305)
        at org.nick.abe.AndroidBackup.extractAsTar(AndroidBackup.java:61)
        ... 1 more

┌──(kali㉿kali)-[~/Desktop]
└─$ ls
abe.jar  test.ab

┌──(kali㉿kali)-[~/Desktop]
└─$ cd ..

┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads                        Music    Public     Videos
Documents  Mobile-Security-Framework-MobSF  Pictures Templates

┌──(kali㉿kali)-[~]
└─$ cd Desktop

┌──(kali㉿kali)-[~/Desktop]
└─$ ls
abe.jar  test.ab

┌──(kali㉿kali)-[~/Desktop]
└─$ java -jar abe.jar unpack test.ab test.tar 1234
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Calculated MK checksum (use UTF-8: true): E016718922013B55BDB5E1614A3F9882950
D3145A04CD2C477D5A235E47F8EE2
13% 26% 34% 35% 36% 37% 38% 39% 53% 66% 79% 86%
72192 bytes written to test.tar.
```



```
┌──(kali㉿kali)-[~/Desktop]
└─$ java -jar abe.jar unpack test.ab test.tar 1234
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Calculated MK checksum (use UTF-8: true): E016718922013B55BDB5E1614A3F9882950
D3145A04CD2C477D5A235E47F8EE2
13% 26% 34% 35% 36% 37% 38% 39% 53% 66% 79% 86%
72192 bytes written to test.tar.

┌──(kali㉿kali)-[~/Desktop]
└─$ tar tvf test.tar
-rw------- 1000/1000       991 1970-01-01 05:30 apps/owasp.sat.agoat/_manifes
t
drwxrwx--x 10060/10060       0 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_te
xtures
drwx------ 10060/10060       0 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview
-rw------- 10060/10060      56 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/pref_store
drwx------ 10060/10060       0 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/GPUCache
drwx------ 10060/10060       0 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/GPUCache/index-dir
-rw------- 10060/10060      48 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/GPUCache/index-dir/the-real-index
-rw------- 10060/10060      20 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/GPUCache/index
drwx------ 10060/10060       0 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/blob_storage
drwx------ 10060/10060       0 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/blob_storage/9fbd3d19-86a1-4102-9a57-d5d28ef052de
-rw------- 10060/10060       0 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/Web Data-journal
-rw------- 10060/10060   57344 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/Web Data
-rw------- 10060/10060      36 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/metrics_guid
-rw------- 10060/10060       0 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/webview_data.lock
-rw------- 10060/10060       0 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/variations_stamp
-rw------- 10060/10060     181 2023-09-29 00:08 apps/owasp.sat.agoat/r/app_we
bview/variations_seed_new
-rw-rw---- 10060/10060     178 2023-09-29 00:09 apps/owasp.sat.agoat/sp/pinDe
tails.xml
-rw-rw---- 10060/10060     127 2023-09-29 00:08 apps/owasp.sat.agoat/sp/WebVi
ewChromiumPrefs.xml

┌──(kali㉿kali)-[~/Desktop]
└─$ tar xvf test.tar
apps/owasp.sat.agoat/_manifest
apps/owasp.sat.agoat/r/app_textures
apps/owasp.sat.agoat/r/app_webview
apps/owasp.sat.agoat/r/app_webview/pref_store
apps/owasp.sat.agoat/r/app_webview/GPUCache
```



2. Sensitive Data Hardcoding:
    - Vulnerable Application – Andro Goat.
    - Vulnerability Description:
      Hard coding sensitive information, such as passwords, server IP addresses, and encryption keys can expose the information to attackers. Anyone who has access to the class files can decompile them and discover the sensitive information.
    - Steps to Reproduce:
        - Step 1: Open Appie and get the source code of the application using apktool.
        - Step 2: Analyse the source code using Java Decompiler.
        - Step 3: I got the promocode from the source code.
        - Step 4: use the promocode to check.
    - Impact:
        - Attackers can get access to sensitive information
        - Attackers can use the information to exploit the app or can be used to gather more information of the app.
    - Proof of Concept:



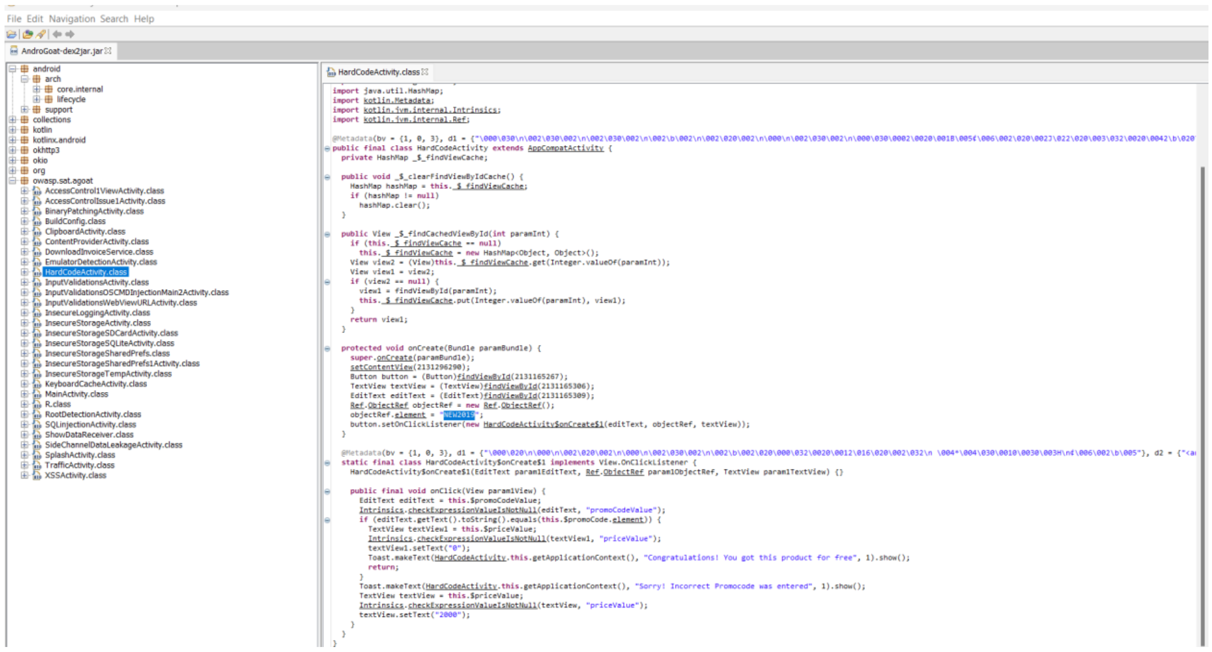
Hardcode Issue

Objectives:
1. Find out how/where Promocode is hardcoded.
2. Enter Promocode to get below product for free

Qty: 1 Price: 2000

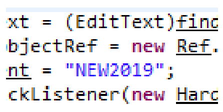
Enter Promocode here

VERIFY

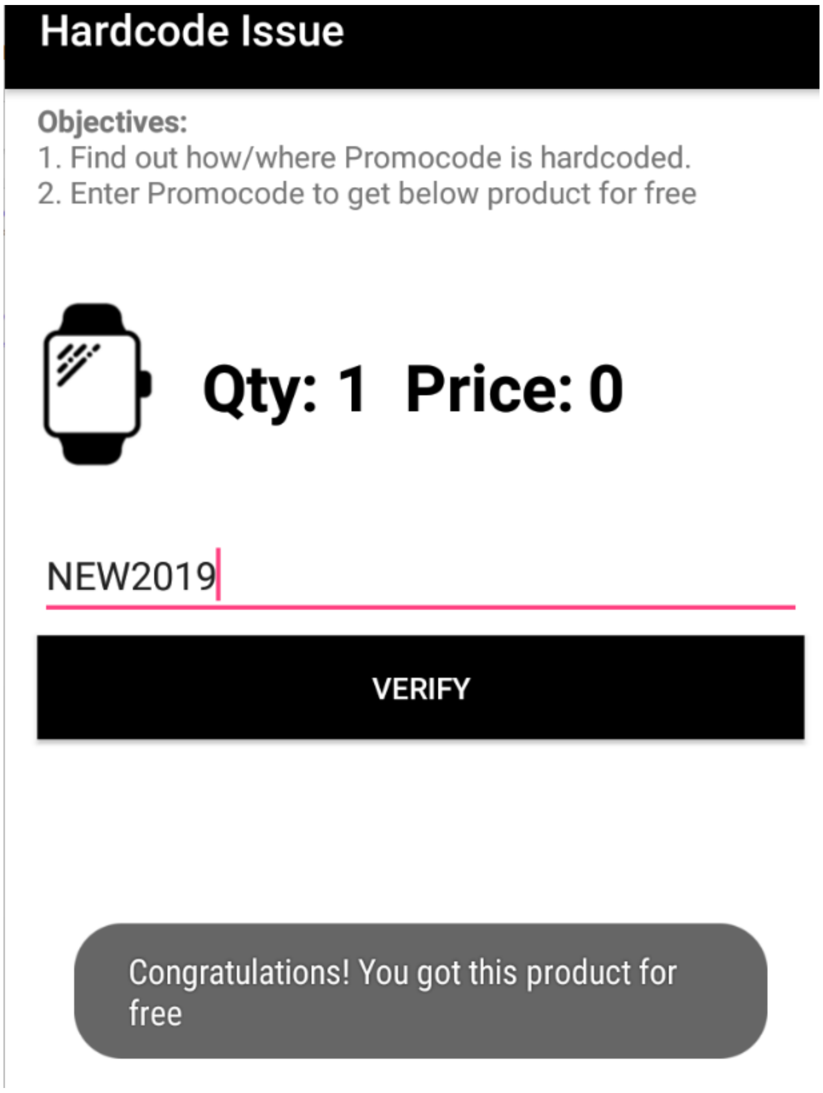


```
android
  arch
    core.internal
    lifecycle
  support
collections
kotlin
kotlinx.android
okhttp3
okio
org
owasp.sat.agoat
  AccessControl1ViewActivity.class
  AccessControlIssue1Activity.class
  BinaryPatchingActivity.class
  BuildConfig.class
  ClipboardActivity.class
  ContentProviderActivity.class
  DownloadInvoiceService.class
  EmulatorDetectionActivity.class
  HardCodeActivity.class
  InputValidationsActivity.class
  InputValidationsOSCMDInjectionMain2Activity.class
  InputValidationsWebViewURLActivity.class
  InsecureLoggingActivity.class
  InsecureStorageActivity.class
  InsecureStorageSDCardActivity.class
  InsecureStorageSQLiteActivity.class
  InsecureStorageSharedPrefs.class
  InsecureStorageSharedPrefs1Activity.class
  InsecureStorageTempActivity.class
  KeyboardCacheActivity.class
  MainActivity.class
  R.class
  RootDetectionActivity.class
  SQLinjectionActivity.class
  ShowDataReceiver.class
  SideChannelDataLeakageActivity.class
  SplashActivity.class
  TrafficActivity.class
  XSSActivity.class
```

```java
import java.util.HashMap;
import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import kotlin.jvm.internal.Ref;

@Metadata(bv = {1, 0, 3}, d1 = {"\000\030\n\002\030\002\n\002\030\002\n\002\b\002\n\002\020\002\n\000\n\002\030\002\n\000\030\0002\0020\001B\005¢\006\002\020\002J\022\020\003\032\0020\0042\b\020
public final class HardCodeActivity extends AppCompatActivity {
  private HashMap _$_findViewCache;

  public void _$_clearFindViewByIdCache() {
    HashMap hashMap = this._$_findViewCache;
    if (hashMap != null)
      hashMap.clear();
  }

  public View _$_findCachedViewById(int paramInt) {
    if (this._$_findViewCache == null)
      this._$_findViewCache = new HashMap<Object, Object>();
    View view2 = (View)this._$_findViewCache.get(Integer.valueOf(paramInt));
    View view1 = view2;
    if (view2 == null) {
      view1 = findViewById(paramInt);
      this._$_findViewCache.put(Integer.valueOf(paramInt), view1);
    }
    return view1;
  }

  protected void onCreate(Bundle paramBundle) {
    super.onCreate(paramBundle);
    setContentView(2131296290);
    Button button = (Button)findViewById(2131165267);
    TextView textView = (TextView)findViewById(2131165306);
    EditText editText = (EditText)findViewById(2131165309);
    Ref.ObjectRef objectRef = new Ref.ObjectRef();
    objectRef.element = "NEW2019";
    button.setOnClickListener(new HardCodeActivity$onCreate$1(editText, objectRef, textView));
  }

  @Metadata(bv = {1, 0, 3}, d1 = {"\000\020\n\000\n\002\020\002\n\000\n\002\030\002\n\002\b\002\020\000\032\0020\0012\016\020\002\032\n \004*\004\030\0010\0030\003H\n¢\006\002\b\005"}, d2 = {"<a
  static final class HardCodeActivity$onCreate$1 implements View.OnClickListener {
    HardCodeActivity$onCreate$1(EditText param1EditText, Ref.ObjectRef param1ObjectRef, TextView param1TextView) {}

    public final void onClick(View param1View) {
      EditText editText = this.$promoCodeValue;
      Intrinsics.checkExpressionValueIsNotNull(editText, "promoCodeValue");
      if (editText.getText().toString().equals(this.$promoCode.element)) {
        TextView textView1 = this.$priceValue;
        Intrinsics.checkExpressionValueIsNotNull(textView1, "priceValue");
        textView1.setText("0");
        Toast.makeText(HardCodeActivity.this.getApplicationContext(), "Congratulations! You got this product for free", 1).show();
        return;
      }
      Toast.makeText(HardCodeActivity.this.getApplicationContext(), "Sorry! Incorrect Promocode was entered", 1).show();
      TextView textView = this.$priceValue;
      Intrinsics.checkExpressionValueIsNotNull(textView, "priceValue");
      textView.setText("2000");
    }
  }
}
```



```
xt = (EditText)finc
bjectRef = new Ref.
nt = "NEW2019";
ckListener(new Harc
```


Hardcode Issue

Objectives:
1. Find out how/where Promocode is hardcoded.
2. Enter Promocode to get below product for free

Qty: 1 Price: 0

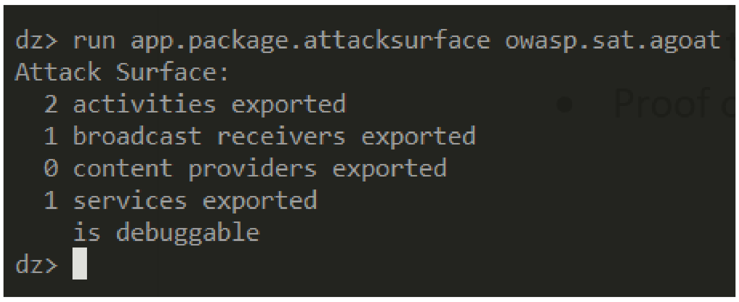
NEW2019

VERIFY

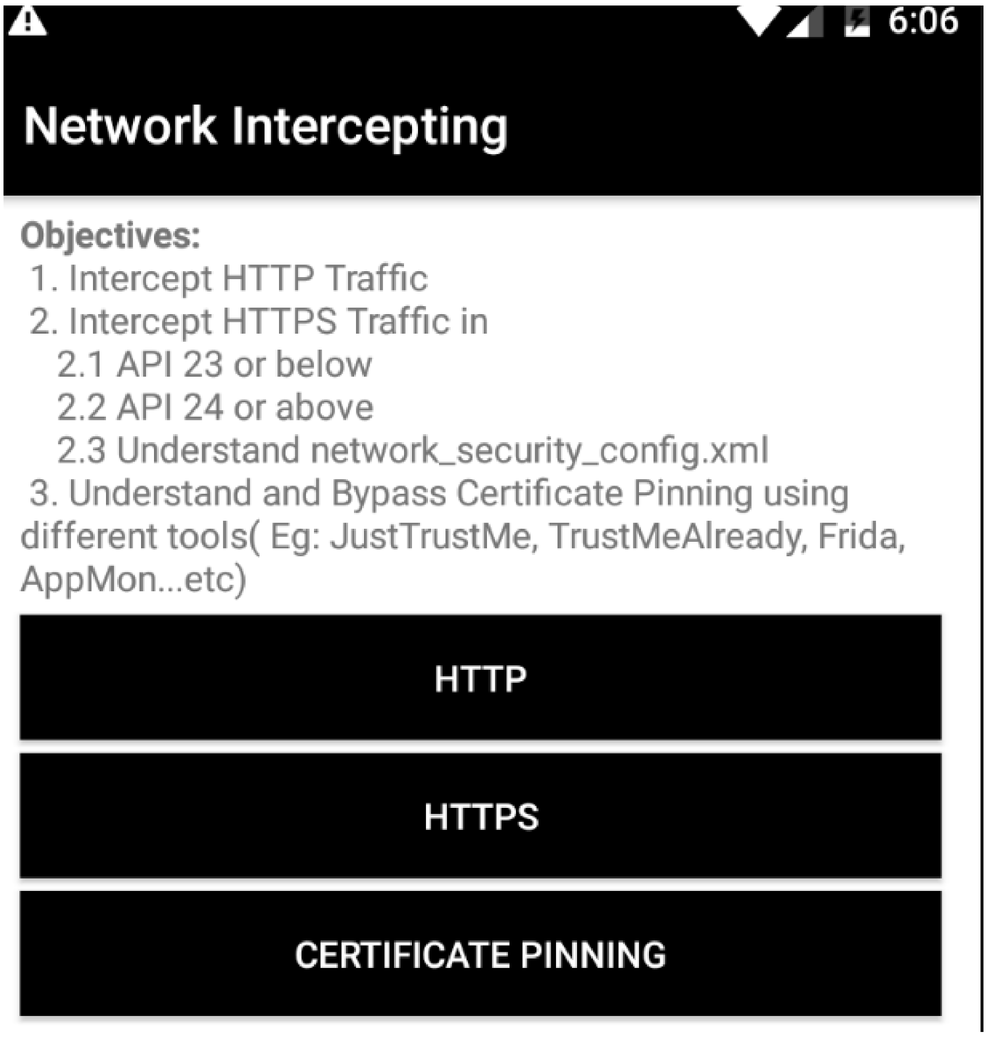
Congratulations! You got this product for free

3. Denial of Service Attack:
   - Vulnerable Application – Andro Goat.
   - Vulnerability Description:
     A denial-of-service (DoS) attack is a cyberattack on devices, information systems, or other network resources that prevents legitimate users from accessing expected services and resources. This is usually accomplished by flooding the targeted host or network with traffic until the target can't respond or crashes.
   - Steps to Reproduce:
     - Step 1: Open Appie and check the attack vector of the app and if the activity is 1 or more then check the activity.
     - Step 2: Then use “run app.activity.start --component owasp.sat.agoat Activity” to do a dos attack.

  - Step 3: DoS attack happened in “run app.activity.start --component owasp.sat.agoat owasp.sat.agoat.Splash Activity” this command.
- Impact:
  - Attackers can stop the app execution.
  - Attackers can crash the app and cause data, money and time loss.
- Proof of Concept:

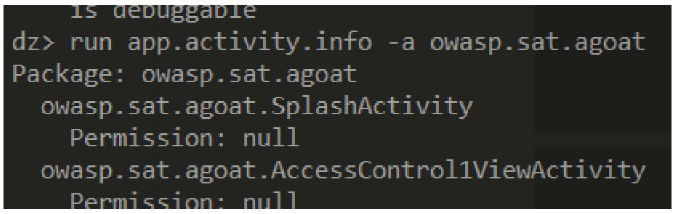


```
dz> run app.package.attacksurface owasp.sat.agoat
Attack Surface:
  2 activities exported
  1 broadcast receivers exported
  0 content providers exported
  1 services exported
    is debuggable
dz>
```

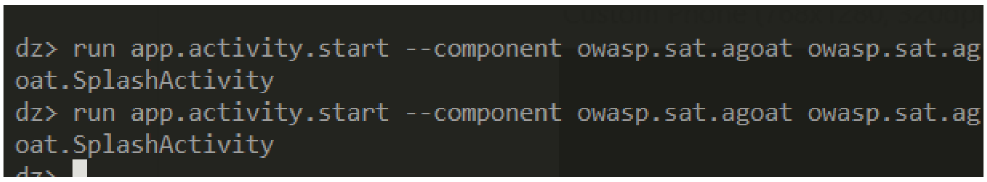


Network Intercepting

Objectives:
1. Intercept HTTP Traffic
2. Intercept HTTPS Traffic in
   2.1 API 23 or below
   2.2 API 24 or above
   2.3 Understand network_security_config.xml
3. Understand and Bypass Certificate Pinning using different tools(Eg: JustTrustMe, TrustMeAlready, Frida, AppMon...etc)

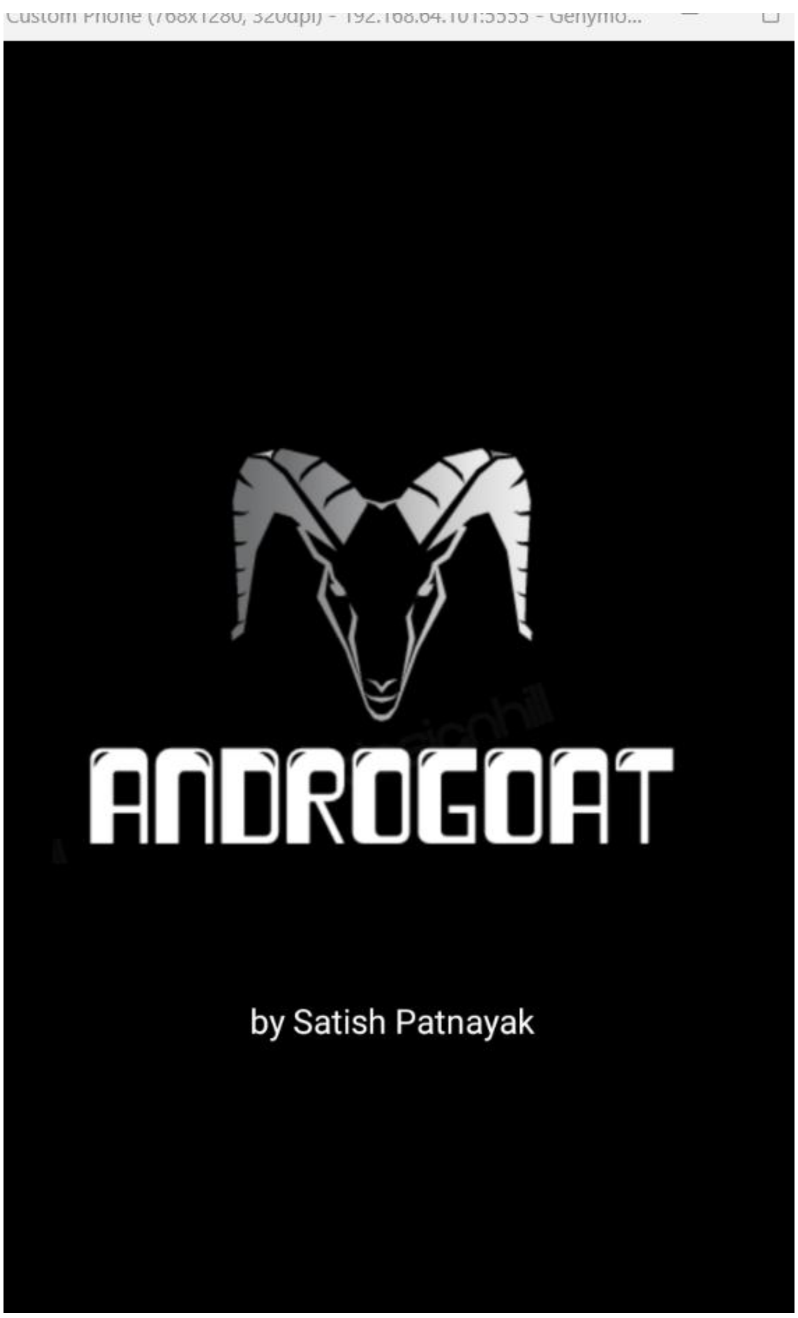


```
    is debuggable
dz> run app.activity.info -a owasp.sat.agoat
Package: owasp.sat.agoat
  owasp.sat.agoat.SplashActivity
    Permission: null
  owasp.sat.agoat.AccessControl1ViewActivity
    Permission: null
```

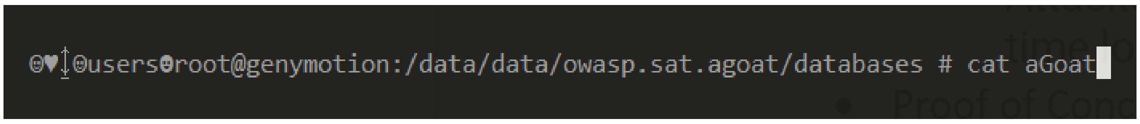


```
dz> run app.activity.start --component owasp.sat.agoat owasp.sat.ag
oat.SplashActivity
dz> run app.activity.start --component owasp.sat.agoat owasp.sat.ag
oat.SplashActivity
dz>
```

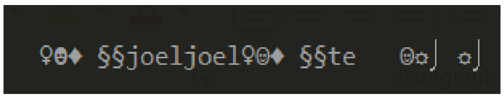
App crashed by the above execution


Custom Phone (768x1280, 320dpi) - 192.168.64.101:5555 - Genymo...
ANDROGOAT
by Satish Patnayak

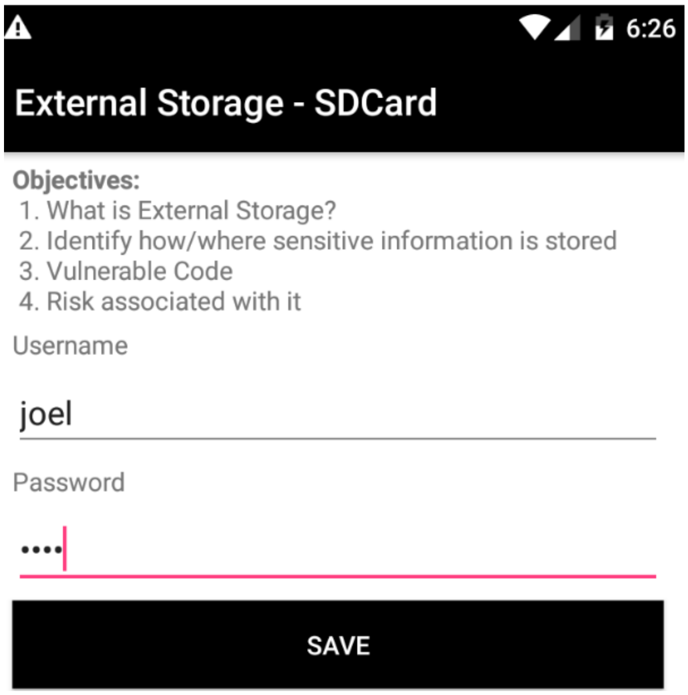
4. Insecure Data Storage:
    - Vulnerable Application – Andro Goat.
    - Vulnerability Description:
      Insecure data storage vulnerabilities occur when development teams assume that users or malware will not have access to a mobile device's filesystem and subsequent sensitive information in data-stores on the device.
    - Steps to Reproduce:
        - Step 1: Open the application and store an information.
        - Step 2: Open Appie and analyse the Android files for the information.
        - Step 3: Got the information in Database/aGoat.
    - Impact:
        - Attackers gain access to sensitive information.
        - Attackers can use the information to exploit the application.
    - Proof of Concept:



```
⊕♥↕⊕users⊕root@genymotion:/data/data/owasp.sat.agoat/databases # cat aGoat
```



```
♀⊕♦ §§joeljoel♀⊕♦ §§te   ⊕¤⌡ ¤⌡
```



6:26

External Storage - SDCard

Objectives:
1. What is External Storage?
2. Identify how/where sensitive information is stored
3. Vulnerable Code
4. Risk associated with it

Username

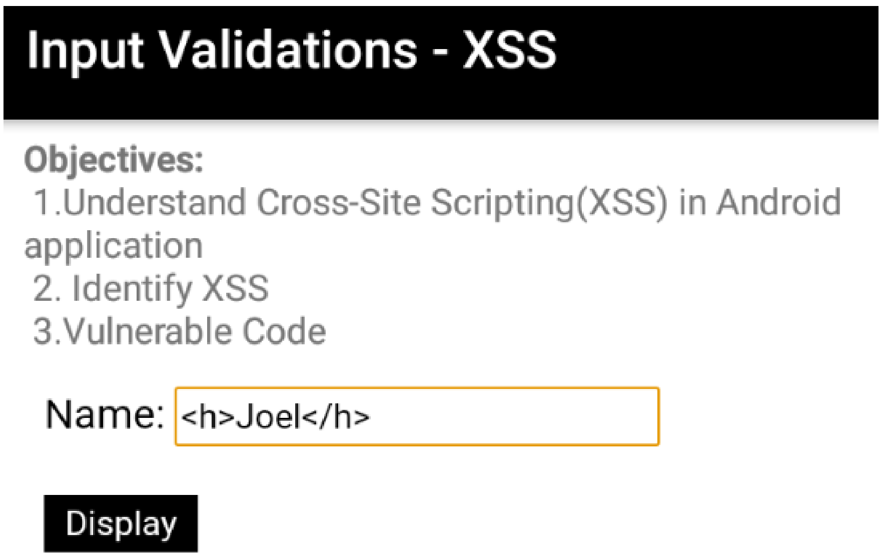
joel

Password

••••

SAVE

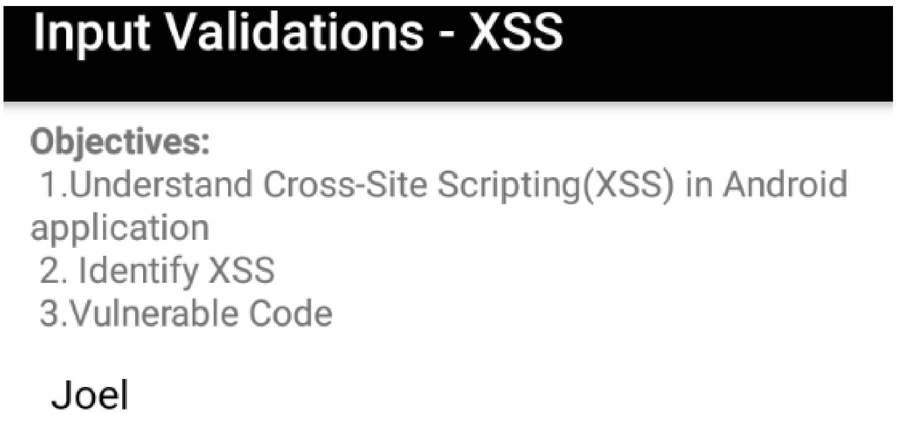
5. Cross Site Scripting (XSS):
   - Vulnerable Application – Andro Goat.
   - Vulnerability Description:
     Cross-site scripting (XSS) is an attack in which an attacker injects malicious executable scripts into the code of a trusted application or website. Attackers often initiate an XSS attack by sending a malicious link to a user and enticing the user to click it.
   - Steps to Reproduce:
     - Step 1: Open the application and try to run html scripts.
     - Step 2: The html script inserted has been executed.
   - Impact:
     - Attackers gain access to sensitive information.
     - Attackers can gain remote access to the system using RCE.
   - Proof of Concept:



Input Validations - XSS

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

Name: <h>Joel</h>



Input Validations - XSS

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

Joel