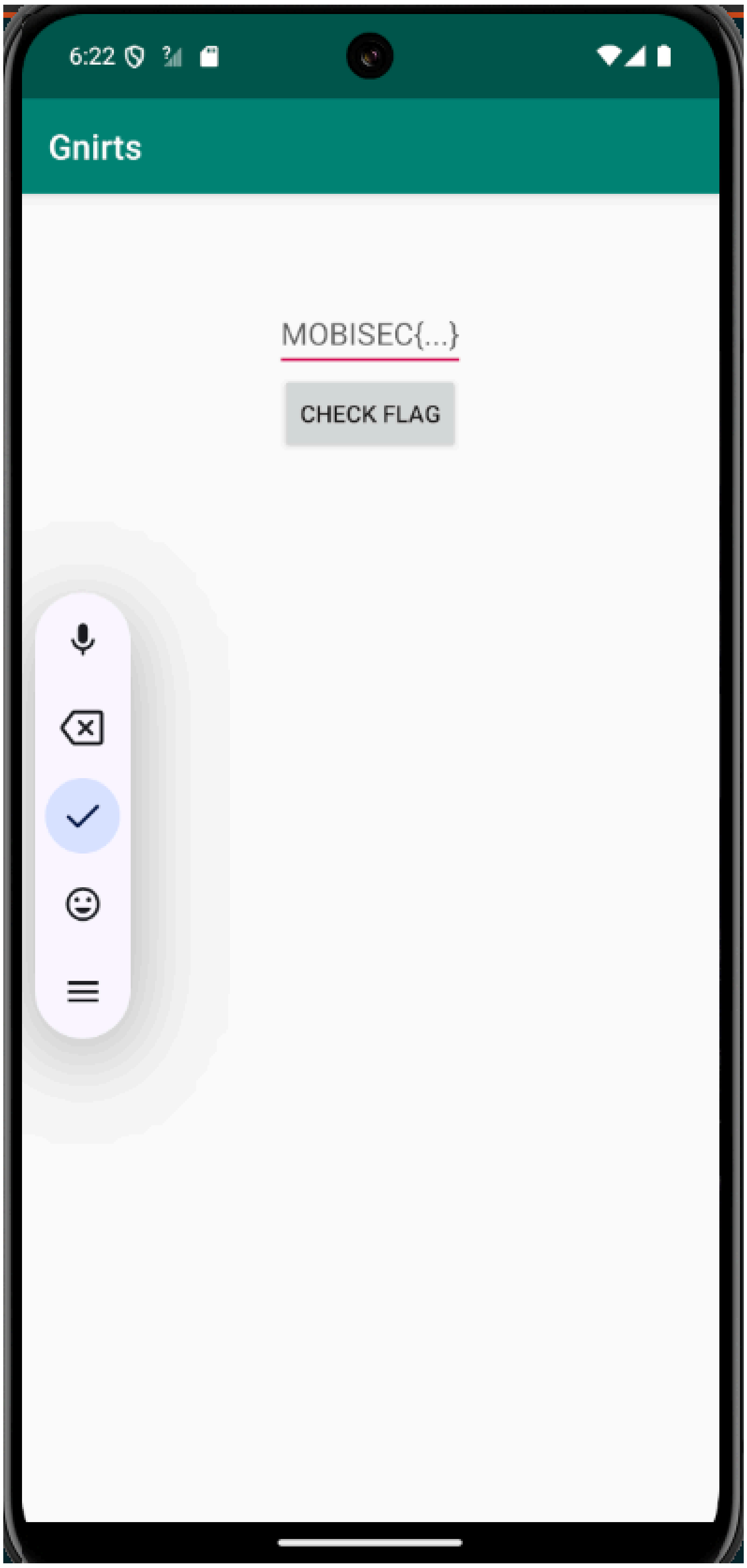
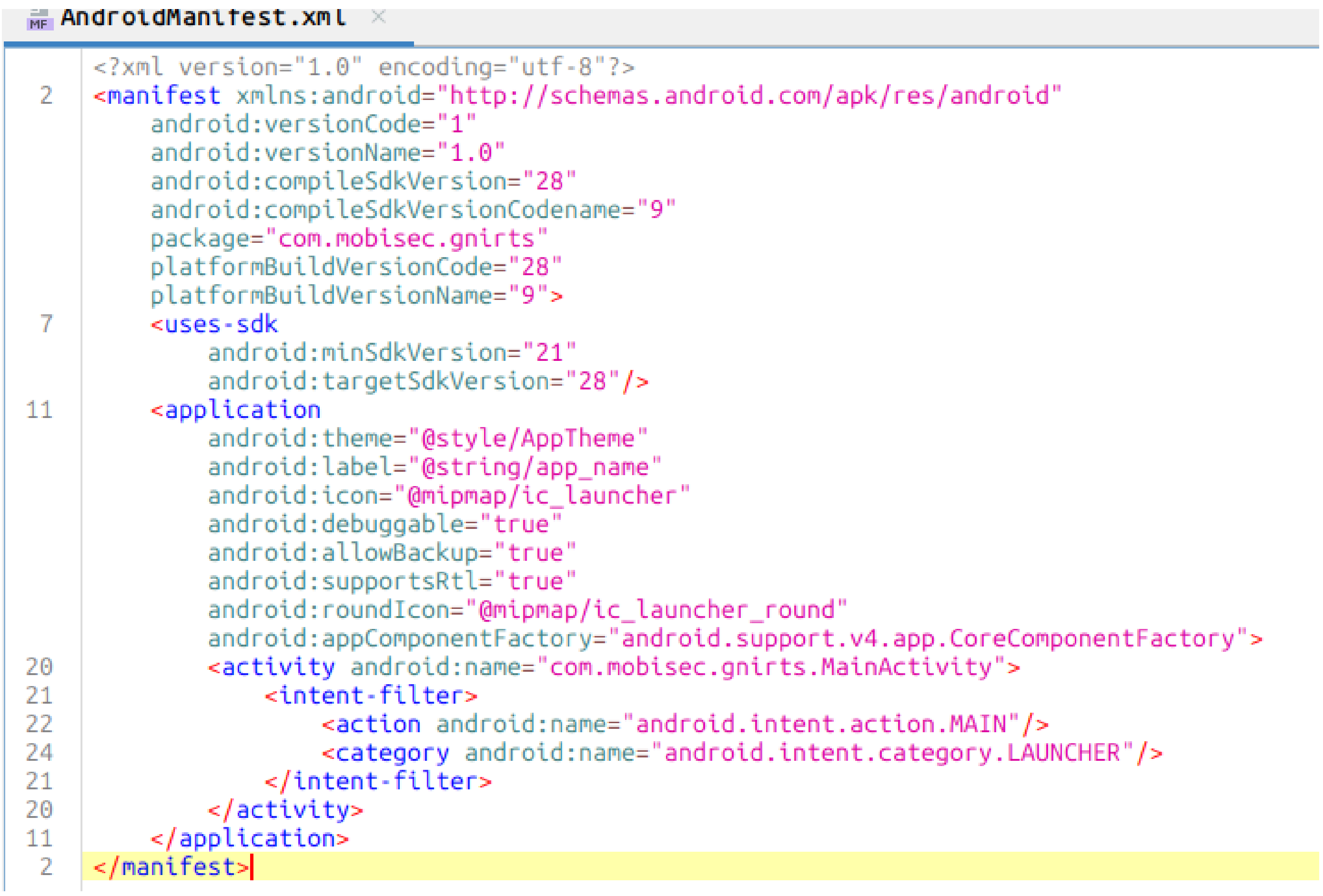
3-Ginarts : reverse engineering

هنا ده بقي 3 CTF معانا وهنا هو طالب ايه هو flag و هيبقي طويل شوية



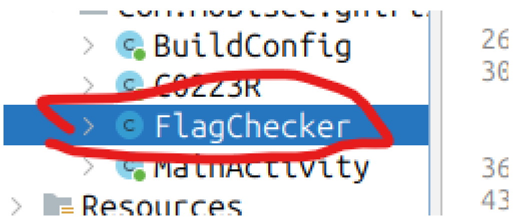
اول حاجة نشوف AndroidManifest.xml علشان لو فيه اي activity مقلا او اي حاجة تانية بس هو مفهوش اي حاجة غير MainActivity



AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    android:compileSdkVersion="28"
    android:compileSdkVersionCodename="9"
    package="com.mobisec.gnirts"
    platformBuildVersionCode="28"
    platformBuildVersionName="9">
    <uses-sdk
        android:minSdkVersion="21"
        android:targetSdkVersion="28"/>
    <application
        android:theme="@style/AppTheme"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:debuggable="true"
        android:allowBackup="true"
        android:supportsRtl="true"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:appComponentFactory="android.support.v4.app.CoreComponentFactory">
        <activity android:name="com.mobisec.gnirts.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

دلوقتي بقي لو روحنا للملفات هنلاقي الفايل اللي المفروض هنحلله علشان نطلع منه flag --->



دلوقتي هنا اه الكود كامل بس هنشوف فيه فانكشن فانكشن ونحاول نفهمه

```
package com.mobisec.gnirts;

import android.content.Context;
import android.util.Base64;
import android.util.Log;
import java.lang.reflect.Method;
import java.security.MessageDigest;
import java.util.HashSet;
import java.util.Set;

/* loaded from: classes.dex */
class FlagChecker {
    FlagChecker() {
    }

    public static boolean checkFlag(Context ctx, String flag) {
        if (!flag.startsWith("MOBISEC{") || !flag.endsWith("}")) {
            return false;
        }
        String core = flag.substring(8, 40);
        if (core.length() != 32) {
            return false;
        }
        String[] ps = core.split(foo());
        if (ps.length != 5 || !bim(ps[0]) || !bum(ps[2]) || !bam(ps[4])) {
            return false;
        }
        String reduced = core.replaceAll("[A-Z]", "X").replaceAll("[a-z]",
"x").replaceAll("[0-9]", " ");
        if (!reduced.matches("[A-Za-z0-9]+.       .[A-Za-z0-9]+.[Xx ]+.[A-
Za-z0-9 ]+")) {
            return false;
        }
        char[] syms = new char[4];
        int[] idxs = {13, 21, 27, 32};
        Set<Character> chars = new HashSet<>();
        for (int i = 0; i < syms.length; i++) {
            syms[i] = flag.charAt(idxs[i]);
            chars.add(Character.valueOf(syms[i]));
        }
        int sum = 0;
        for (char c : syms) {
            sum += c;
        }
        return sum == 180 && chars.size() == 1 && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct1),
ctx.getString(C0223R.string.f22k1)), ps[0]),
ctx.getString(C0223R.string.f29t1)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct2),
ctx.getString(C0223R.string.f23k2)), ps[1]),
ctx.getString(C0223R.string.f30t2)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct3),
ctx.getString(C0223R.string.f24k3)), ps[2]),
ctx.getString(C0223R.string.f31t3)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct4),
ctx.getString(C0223R.string.f25k4)), ps[3]),
ctx.getString(C0223R.string.f32t4)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct5),
ctx.getString(C0223R.string.f26k5)), ps[4]),
ctx.getString(C0223R.string.f33t5)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct6),
ctx.getString(C0223R.string.f27k6)), flag),
ctx.getString(C0223R.string.f34t6));
    }

    private static boolean bim(String s) {
        return s.matches("^[a-z]+$");
    }

    private static boolean bum(String s) {
        return s.matches("^[A-Z]+$");
    }

    private static boolean bam(String s) {
        return s.matches("^[0-9]+$");
    }

    /* renamed from: dh */
    private static String m8dh(String hash, String s) {
        try {
            MessageDigest md = MessageDigest.getInstance(hash);
            md.update(s.getBytes());
            byte[] digest = md.digest();
            return toHexString(digest);
        } catch (Exception e) {
            return null;
        }
    }

    private static String toHexString(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            String hex = Integer.toHexString(b & 255);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        return hexString.toString();
    }

    public static String foo() {
        String s =
"Vm0wd2QyVkZNVWRYV0docFVtMVNWVmx0ZEhkVlZscDBUVlpPVmsxWGVIbFdiVFZyVm0xS1IyTkl
iRmRXTTFKTVZsVmFWMVpWTVVWaGVqQTk=";
        for (int i = 0; i < 10; i++) {
            s = new String(Base64.decode(s, 0));
        }
        return s;
    }

    /* renamed from: gs */
    private static String m9gs(String a, String b) {
        String s = BuildConfig.FLAVOR;
        for (int i = 0; i < a.length(); i++) {
            s = s + Character.toString((char) (a.charAt(i) ^ b.charAt(i %
b.length())));
        }
        return s;
    }

    /* renamed from: me */
    private static boolean m10me(Context ctx, String s1, String s2) {
        try {
            Class c = s1.getClass();
            Method m = c.getMethod(m11r(ctx.getString(C0223R.string.f28m1)),
Object.class);
            boolean res = ((Boolean) m.invoke(s1, s2)).booleanValue();
            return res;
        } catch (Exception e) {
            Log.e("MOBISEC", "Exception: " + Log.getStackTraceString(e));
            return false;
        }
    }

    /* renamed from: r */
    public static String m11r(String s) {
        return new StringBuffer(s).reverse().toString();
    }
}
```

1- Check the flag start with --> **MOBISC{** and End with **}**

```
  public static boolean checkFlag(Context ctx, String flag) {
        if (!flag.startsWith("MOBISEC{") || !flag.endsWith("}")) {
            return false;

```

2- check the size of flag between **{ } == 32**

```
String core = flag.substring(8, 40);
        if (core.length() != 32) {
            return false;
        }
```

3- دلوقتي هنا المفروض نشوف الفانكشن الل بعدها بس هنروح اول حاجة نشوف فانكشن اللي هي foo()

3-Check foo() function

take base64 string and decode it for 10 times

```
public static String foo() {
        String s =
"Vm0wd2QyVkZNVWRYV0docFVtMVNWVmx0ZEhkVlZscDBUVlpPVmsxWGVIbFdiVFZyVm0xS1IyTkl
iRmRXTTFKTVZsVmFWMVpWTVVWaGVqQTk=";
        for (int i = 0; i < 10; i++) {
            s = new String(Base64.decode(s, 0));
        }
        return s;
    }
```

دلوقتي بقي بما ان دي عايزه 10 مرات تعمل decode base64 هنعمل كود python علشان نشوف ايه هو الناتج decode

4-Code python for function foo and the result is --> this character **-**

```
import base64
def foo():

s="Vm0wd2QyVkZNVWRYV0docFVtMVNWVmx0ZEhkVlZscDBUVlpPVmsxWGVIbFdiVFZyVm0xS1IyT
kliRmRXTTFKTVZsVmFWMVpWTVVWaGVqQTk="
    for i in range(0,10):
        s=base64.b64decode(s).decode('utf-8')
    print("the base64 decode for 10 times is : ",s)
foo()

the result is -->          -
```

هنكمل بقي دلوقتي عادي لما نكمل هنلاقي ان فانكشن اللي هي split(foo()) دي هنلاقيها بتحول من string to array بس بتحذف ده -

5- ps variable covert flag between {} from string to array and remove this char - and the result ps will contain array from 5 word

from ps[0] to ps[4]

```
String[] ps = core.split(foo());
        if (ps.length != 5 || !bim(ps[0]) || !bum(ps[2]) || !bam(ps[4])) {
            return false;
        }
```

now we know the flag will be MOBISEC{ps[0]-ps[1]-ps[2]-ps[3]-ps[4])

دلوقتي بقي عاوزين نشوف ايه فانكش اللي هما bim , bum and bam

6-Check function bim, bum and bam

- **bim --- match (a-z)**
- **bum ---> match (A-Z)**
- **bam ---> match (0-9)**

كده بقي في الخطوة رقم 5

- **p[0] --> match (a-z)**
- **p[2] --> match (A-Z)**
- **p4 ---> match (0-9)**

```
private static boolean bim(String s) {
        return s.matches("^[a-z]+$");
    }

    private static boolean bum(String s) {
        return s.matches("^[A-Z]+$");
    }

    private static boolean bam(String s) {
        return s.matches("^[0-9]+$");
    }
```

now the flag match MOBSIEC{(a-z)-ps[1]-(A-Z)-ps[3]-(0-9)}

دلوقتي بقي بعد كده هو هيشرح اكتر ازاي flag match

7-check the word in flag match

- **ps[0] --> match (a-z)**
- **ps[1] --> match numbers (0-9)**
- **ps[2] --> match (A-Z)**
- **ps[3] ---> match numbers and a-z and A-Z**
- **ps[4] --> match number (0-9)**

```
String reduced = core.replaceAll("[A-Z]", "X").replaceAll("[a-z]",
"x").replaceAll("[0-9]", " ");
        if (!reduced.matches("[A-Za-z0-9]+.       .[A-Za-z0-9]+.[Xx ]+.[A-
Za-z0-9 ]+")) {
            return false;
        }
```

دلوقتي بقي هو هيعرفنا علي مكان الحرف اللي بيفصل بين الكلمات اللي هو ده - وبعد كده هيستدعي شوية vaiable من التطبيق نفسة اللي هيكونو لو عملنا decompile for app هيبقو في gnirts/res/values/strings.xml/

8- this is the most part in code

دلوقتي زي ما قولنا هنا بيحدد مكان - والقيمة اللي بيعملها return

return sum180 && chars.size() 1 ---> mean chars contain 4 char are same value so the size is 1 and the sum of these chars is 180 mean the char for ascii --> 180/4 =45 --> -

now flag --> MOBISEC{XXXX-XXXXXXX-XXXXX-XXXX-XXXXXXX}

```
char[] syms = new char[4];
        int[] idxs = {13, 21, 27, 32};
        Set<Character> chars = new HashSet<>();
        for (int i = 0; i < syms.length; i++) {
            syms[i] = flag.charAt(idxs[i]);
            chars.add(Character.valueOf(syms[i]));
        }
        int sum = 0;
        for (char c : syms) {
            sum += c;
        }
        return sum == 180 && chars.size() == 1 && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct1),
ctx.getString(C0223R.string.f22k1)), ps[0]),
ctx.getString(C0223R.string.f29t1)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct2),
ctx.getString(C0223R.string.f23k2)), ps[1]),
ctx.getString(C0223R.string.f30t2)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct3),
ctx.getString(C0223R.string.f24k3)), ps[2]),
ctx.getString(C0223R.string.f31t3)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct4),
ctx.getString(C0223R.string.f25k4)), ps[3]),
ctx.getString(C0223R.string.f32t4)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct5),
ctx.getString(C0223R.string.f26k5)), ps[4]),
ctx.getString(C0223R.string.f33t5)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct6),
ctx.getString(C0223R.string.f27k6)), flag),
ctx.getString(C0223R.string.f34t6));
    }
```

دلوقتي بقي قولنا ان ده اهم جزء علشان ده اللي من خلاله هنعرف flag وده في function 3 و القيم اللي هجيبها اول حاجة ان هجشب القيم من الملف وبع كده نشرح الفانكشن

```
        <string name="ct1">xwe</string>
        <string name="ct2">asd</string>
        <string name="ct3">uyt</string>
        <string name="ct4">42s</string>
        <string name="ct5">p0X</string>
        <string name="ct6">70 IJTR</string>
        <string name="k1">53P</string>
        <string name="k2">,7Q</string>
        <string name="k3">8=A</string>
        <string name="k4">yvF</string>
        <string name="k5">=tm</string>
        <string name="k6">dxa</string>
        <string name="m1">slauqe</string>
        <string name="t1">6e9a4d130a9b316e9201238844dd5124</string>
        <string name="t2">7c51a5e6ea3214af970a86df89793b19</string>
        <string name="t3">e5f20324ae520a11a86c7602e29ecbb8</string>
        <string name="t4">1885eca5a40bc32d5e1bca61fcd308a5</string>
        <string name="t5">da5062d64347e5e020c5419cebd149a2</string>
        <string
name="t6">1c4d1410a4071880411f02ff46370e46b464ab2f87e8a487a09e13040d64e396</
string>
```

دلوقتي بقي هنشرح علي السريع ملخص function

9- explain function m8dh

this function take hash type and string s and return the value of string by using hash and convert to hexa

return toHexString(hash(s))

```
private static String m8dh(String hash, String s) {
        try {
            MessageDigest md = MessageDigest.getInstance(hash);
            md.update(s.getBytes());
            byte[] digest = md.digest();
            return toHexString(digest);
        } catch (Exception e) {
            return null;
        }
```

10- explain function m9gs

this function take two string and return the result ov XOR between two string so we will create python code for do this operation

```
 private static String m9gs(String a, String b) {
        String s = BuildConfig.FLAVOR;
        for (int i = 0; i < a.length(); i++) {
            s = s + Character.toString((char) (a.charAt(i) ^ b.charAt(i %
b.length())));
        }
        return s;
    }
```

11-pytho code for XOR

```
def m9():
    arr1=['xwe','asd','uyt','42s','p0X','70 IJTR']
    arr2=['53p',',7Q','8=A','yvF','=tm','dxa']
    arr1_size=len(arr1)
    for x in range(0,arr1_size):
        result = ""
        ct=arr1[x]
        k=arr2[x]
        ks_size = len(k)
        ct_size = len(ct)
        for i in range(ct_size):
            result += chr(ord(ct[i]) ^ ord(k[i % ks_size]))

        print(f"The hash type of line {x+1} is {result}")


m9()
```

```
the result is
The hash type of line 1 is MD
The hash type of line 2 is MD5
The hash type of line 3 is MD5
The hash type of line 4 is MD5
The hash type of line 5 is MD5
The hash type of line 6 is SHA-256
```

12- explain the m10me

this function take two string and return true if s1=s2 else return false

```
    private static boolean m10me(Context ctx, String s1, String s2) {
        try {
            Class c = s1.getClass();
            Method m = c.getMethod(m11r(ctx.getString(C0223R.string.f28m1)),
Object.class);
            boolean res = ((Boolean) m.invoke(s1, s2)).booleanValue();
            return res;
        } catch (Exception e) {
            Log.e("MOBISEC", "Exception: " + Log.getStackTraceString(e));
            return false;
        }
```

بس هنا هنلاقي فانشكن الي هي m11r ودي هنستخدمها علشان نعكس ونحول الفية دي C0223R.string.f28m1 ل string

13-expalin m11r function

this function take value and reverse it and convert to string and this value is

```
<string name="m1">slauqe</string>      in reverse  euqals
```

code for m11r

```
    public static String m11r(String s) {
        return new StringBuffer(s).reverse().toString();
    }
```

دلوقتي هشنرح اهم سطر اللي قولنا عليه ده اهم حاجة اللي من خلاله هنعرف flag

14- how to get the flag

```
        return sum == 180 && chars.size() == 1 && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct1),
ctx.getString(C0223R.string.f22k1)), ps[0]),
ctx.getString(C0223R.string.f29t1)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct2),
ctx.getString(C0223R.string.f23k2)), ps[1]),
ctx.getString(C0223R.string.f30t2)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct3),
ctx.getString(C0223R.string.f24k3)), ps[2]),
ctx.getString(C0223R.string.f31t3)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct4),
ctx.getString(C0223R.string.f25k4)), ps[3]),
ctx.getString(C0223R.string.f32t4)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct5),
ctx.getString(C0223R.string.f26k5)), ps[4]),
ctx.getString(C0223R.string.f33t5)) && m10me(ctx,
m8dh(m9gs(ctx.getString(C0223R.string.ct6),
ctx.getString(C0223R.string.f27k6)), flag),
ctx.getString(C0223R.string.f34t6));
```

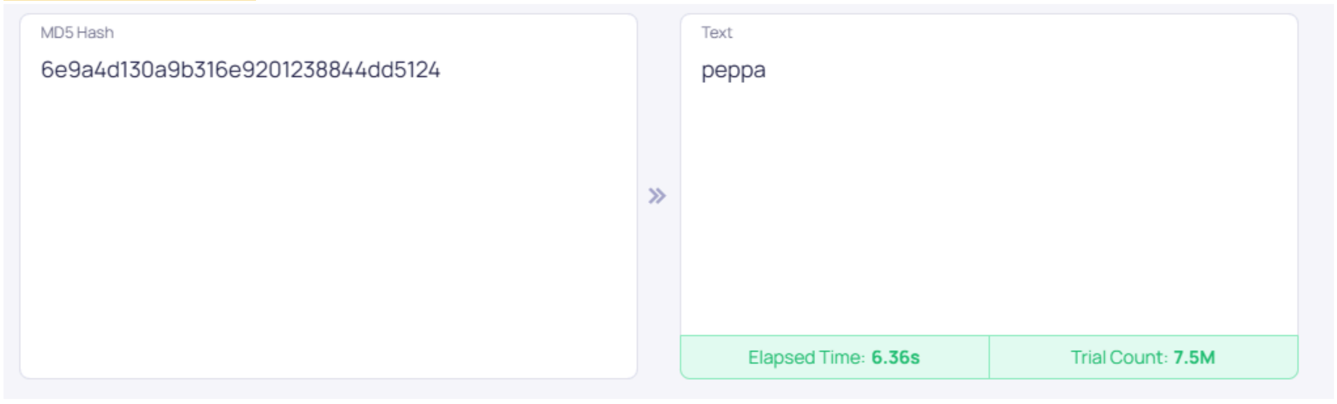
and the summery of this line

```
    // return
    // sum == 180 && chars.size() == 1 &&
    // m10(ctx,m8(m9(xwe,53p),ps[0]),6e9a4d130a9b316e9201238844dd5124) && //
t1
    // m10(ctx,m8(m9(asd,,7Q),ps[1]),7c51a5e6ea3214af970a86df89793b19) && //
t2
    // m10(ctx,m8(m9(uyt,8=A),ps[2]),e5f20324ae520a11a86c7602e29ecbb8) && //
t3
    // m10(ctx,m8(m9(42s,yvF),ps[3]),1885eca5a40bc32d5e1bca61fcd308a5) && //
t4
    // m10(ctx,m8(m9(p0X,=tm),ps[4]),da5062d64347e5e020c5419cebd149a2) && //
t5
    // m10(ctx,m8(m9(70
IJTR,dxa),flag),1c4d1410a4071880411f02ff46370e46b464ab2f87e8a487a09e13040d64
e396) &&
```

15- crack each hash from t1 to t5

- **t1 is peppa**

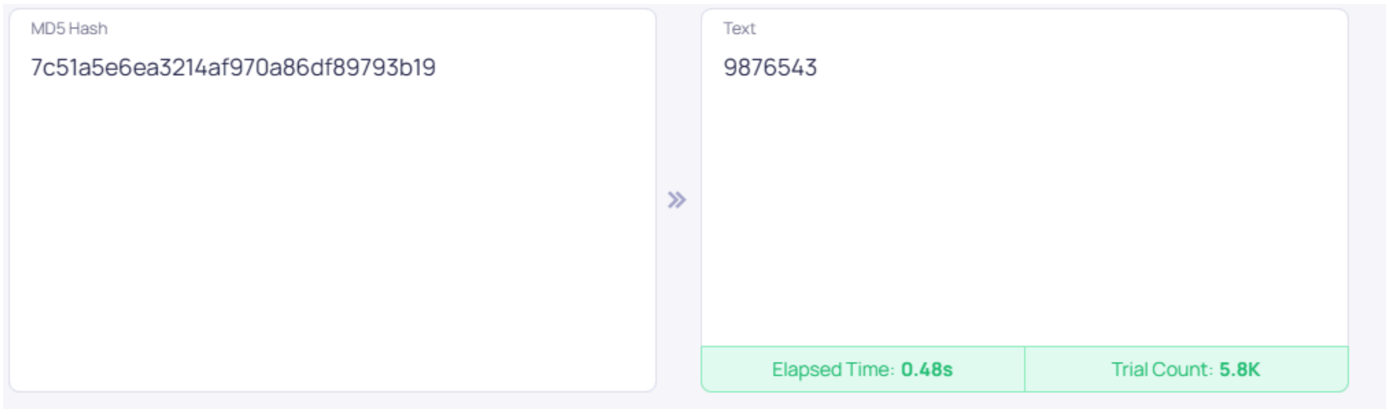
- **ps[0] is peppa**



6e9a4d130a9b316e9201238844dd5124

peppa

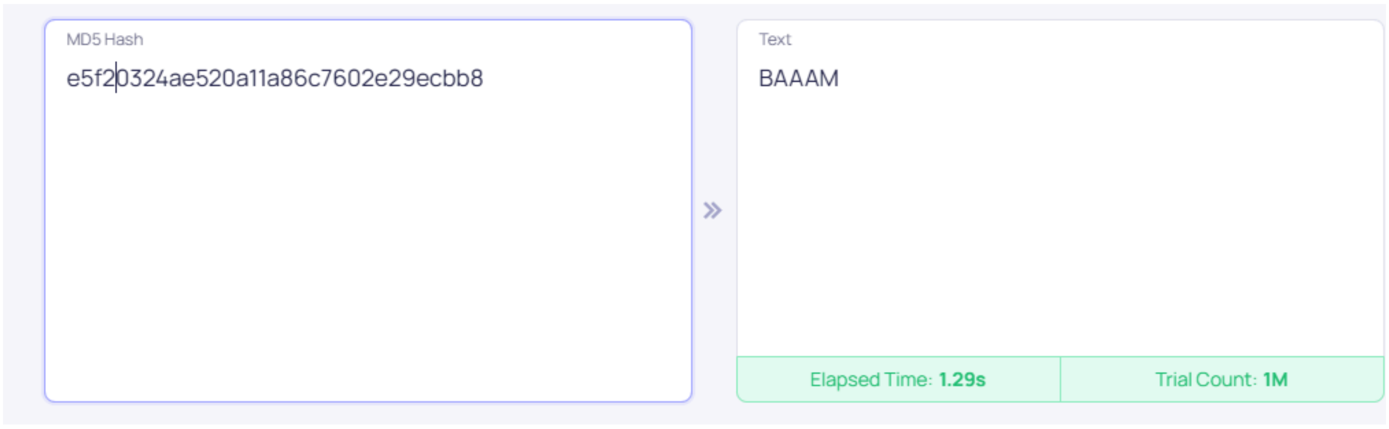
- **t2 is 9876543**
- **ps[1] is 9876543**



7c51a5e6ea3214af970a86df89793b19

9876543

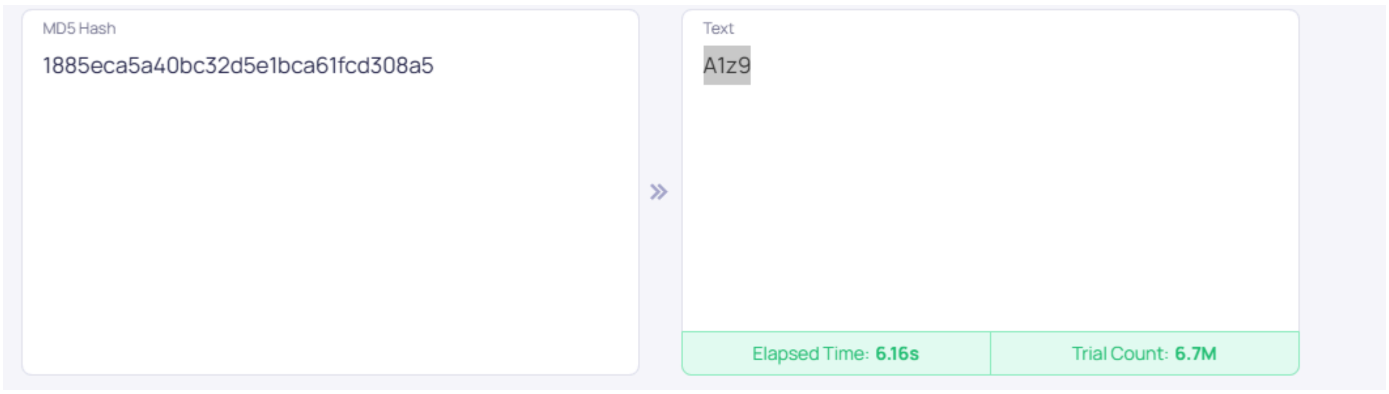
- **t3 is BAAAM**
- **ps[2] is BAAAM**



e5f20324ae520a11a86c7602e29ecbb8

BAAAM

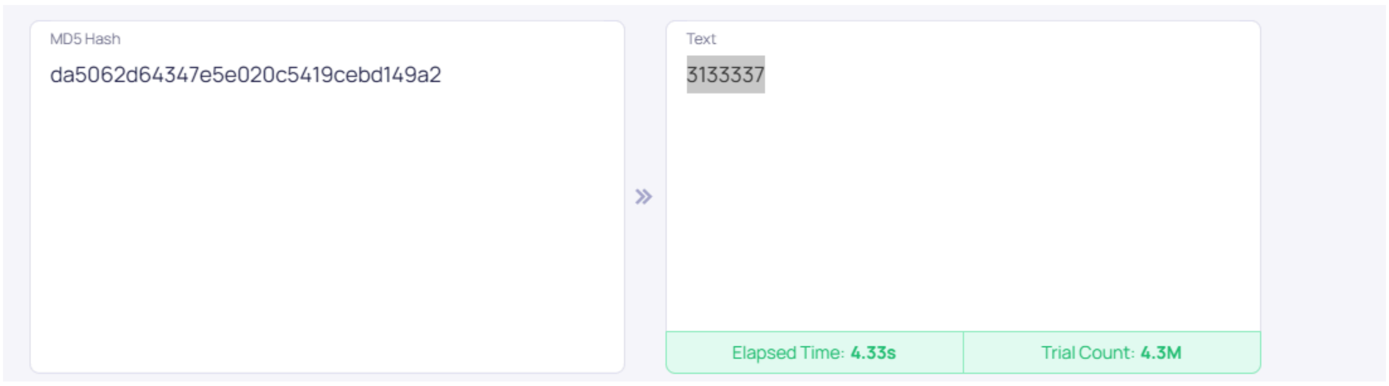
- **t4 is A1z9**
- **ps[3] is A1z9**


MD5 Hash
1885eca5a40bc32d5e1bca61fcd308a5

Text
A1z9

Elapsed Time: 6.16s | Trial Count: 6.7M

- **t5 is 3133337**
- **ps[4] is 3133337**


MD5 Hash
da5062d64347e5e020c5419cebd149a2

Text
3133337

Elapsed Time: 4.33s | Trial Count: 4.3M

finally the flag is MOBISEC{peppa-9876543-BAAAM-A1z9-3133337} and check he equals the sha-256 of values we got it from file on app

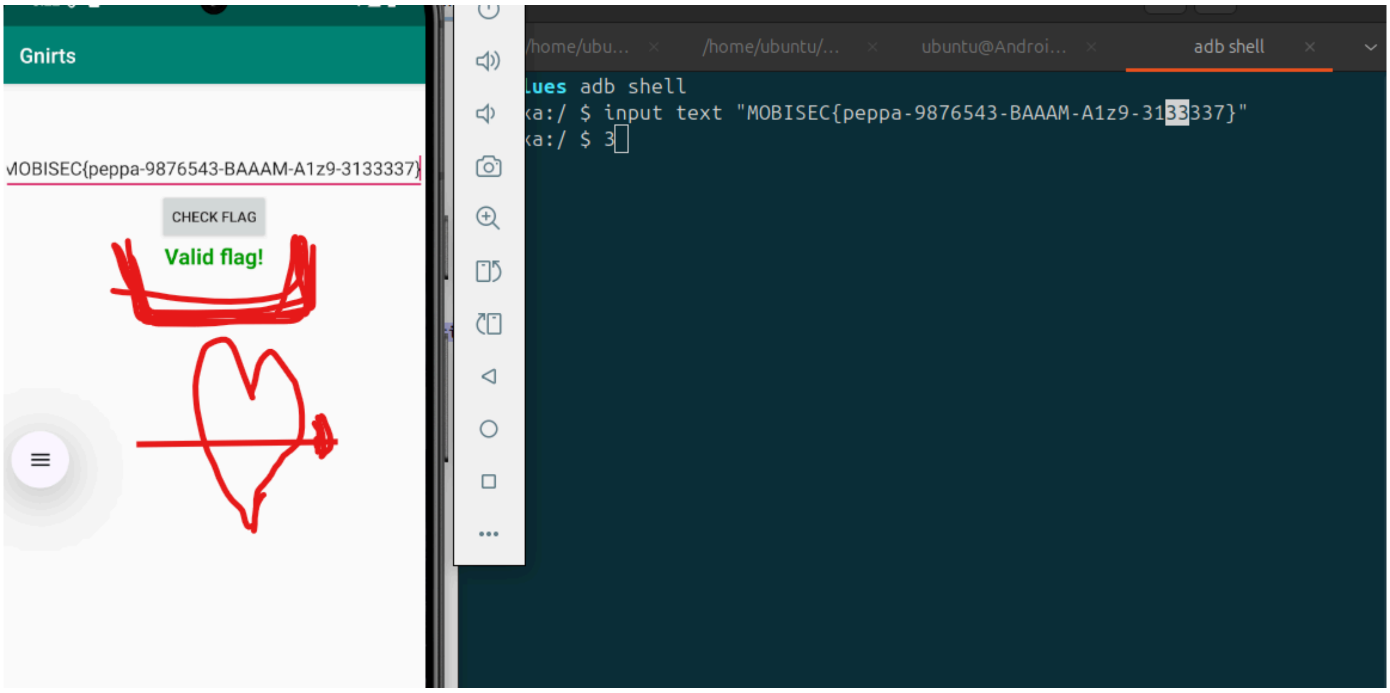
pooom > he equal the value

```
MOBISEC{peppa-9876543-BAAAM-A1z9-3133337}
```

Output

```
1c4d1410a4071880411f02ff46370e46b464ab2f87e8a487a09e13040d64e396
```

check the flag on app



```
lues adb shell
xa:/ $ input text "MOBISEC{peppa-9876543-BAAAM-A1z9-3133337}"
xa:/ $ 3
```