DEKRA

App
com.protectstar.antispy.android
MASA L1
2025-03-06

Summary of the application

Target of Evaluation	**App Version: 6.6.5**
Model and/or type reference	**com.protectstar.antispy.android**
Other identification of the product	**com.protectstar.antispy.android**
Features	**Tools & Utilities**
Manufacturer	**Protectstar Inc.**
Test Method Requested	**Security evaluation based on limited set of evaluation procedures from OWASP Mobile Application Security Verification Standard established by ADA.**
Validation Type	**Level 1 - Verified Self**
Validated By	**Jose María Santos López – Cybersecurity Engineer**
Platform	**Android**
Date of Issue	**2025-03-06**

DEKRA Testing and Certification guarantees the reliability of the data presented in this report, which is the result of the measurements and the tests performed to the item under test on the date and under the conditions stated on the report and, it is based on the knowledge and technical facilities available at DEKRA Testing and Certification at the time of performance of the test.

DEKRA Testing and Certification is liable to the client for the maintenance of the confidentiality of all information related to the item under test and the results of the test. The results presented in this Test Report apply only to the particular item under test established in this document.

Nomenclature

Below you can find the verification type considered for MASA L1 as well as the description of each of them.

Verification Type

- **Self Declare:** developer provides Yes/No response to verify they meet the requirement
- **Documentation:** developer provides artifacts / evidence to verify they meet the requirement
- **NMI:** lab provides output from static analysis for developers to respond to
- **Scan Verified:** lab performs static analysis against fully automatable requirements to assign a Pass / Fail for each requirement
    - For scan failures the developer is expected to resolve the issue or provide a written justification explaining how they meet the requirement including supporting evidence such as scan output, screenshots or supporting documentation.

Summary Table

Requirement	Description	Validation Type	Status	Dev Action
MSTG-STORAGE-1	System credential storage facilities need to be used to store sensitive data, such as PII, user credentials or cryptographic keys	Self Declare	Pass	N
MSTG-STORAGE-2	No sensitive data should be stored outside of the app container or system credential storage facilities.	Self Declare	Pass	N
MSTG-STORAGE-3	No sensitive data is written to application logs.	NMI	Pass	N
MSTG-STORAGE-5	The keyboard cache is disabled on text inputs that process sensitive	Scan Verified	Pass	N

	data.			
MSTG-STORAGE-7	No sensitive data, such as passwords or pins, is exposed through the user interface.	NMI	Pass	N
MSTG-STORAGE-12	The app educates the user about the types of personally identifiable information processed, as well as security best practices the user should follow in using the app.	Self Declare	Pass	N
MSTG-CRYPTO-1	The app does not rely on symmetric cryptography with hardcoded keys as a sole method of encryption.	Scan Verified	Pass	N
MSTG-CRYPTO-2	The app uses proven implementations of cryptographic primitives.	Scan Verified	Pass	N
MSTG-CRYPTO-3	The app uses cryptographic primitives that are appropriate for the particular use-case, configured with parameters that adhere to industry best practices.	Scan Verified	Pass	N
MSTG-CRYPTO-4	The app does not use cryptographic protocols or algorithms that are widely considered deprecated for security purposes.	Scan Verified	Pass	N

MSTG-CRYPTO-5	The app does not re-use the same cryptographic key for multiple purposes.	Self Declare	Pass	N
MSTG-CRYPTO-6	All random values are generated using a sufficiently secure random number generator.	Scan Verified	Pass	N
MSTG-AUTH-1	If the app provides users access to a remote service, some form of authentication, such as username/password authentication, is performed at the remote endpoint.	Self Declare	Pass	N
MSTG-AUTH-2	If stateful session management is used, the remote endpoint uses randomly generated session identifiers to authenticate client requests without sending the user`s credentials.	Self Declare	Pass	N
MSTG-AUTH-3	If stateless token-based authentication is used, the server provides a token that has been signed using a secure algorithm.	Self Declare	Pass	N
MSTG-AUTH-4	The remote endpoint terminates the existing session when the user logs	Self Declare	Pass	N

	out.			
MSTG-AUTH-5	A password policy exists and is enforced at the remote endpoint.	Self Declare	Pass	N
MSTG-AUTH-6	The remote endpoint implements a mechanism to protect against the submission of credentials an excessive number of times.	Self Declare	Pass	N
MSTG-AUTH-7	Sessions are invalidated at the remote endpoint after a predefined period of inactivity and access tokens expire.	Self Declare	Pass	N
MSTG-NETWORK-1	Data is encrypted on the network using TLS. The secure channel is used consistently throughout the app.	Scan Verified	Pass	N
MSTG-NETWORK-2	The TLS settings are in line with current best practices, or as close as possible if the mobile operating system does not support the recommended standards.	Scan Verified	Pass	N
MSTG-NETWORK-3	The app verifies the X.509 certificate of the remote endpoint when the secure channel is established.	Scan Verified	Pass	N

MSTG-PLATFORM-1	The app only requests the minimum set of permissions necessary.	Scan Verified	Pass	N
MSTG-PLATFORM-2	All inputs from external sources and the user are validated and if necessary sanitized. This includes data received via the UI, IPC mechanisms such as intents, custom URLs, and network sources.	Scan Verified	Pass	N
MSTG-PLATFORM-3	The app does not export sensitive functionality via custom URL schemes, unless these mechanisms are properly protected.	Scan Verified	Pass	N
MSTG-PLATFORM-4	The app does not export sensitive functionality through IPC facilities, unless these mechanisms are properly protected.	Self Declare	Pass	N
MSTG-CODE-1	The app is signed and provisioned with a valid certificate, of which the private key is properly protected.	Scan Verified	Pass	N
MSTG-CODE-2	The app has been built in release mode, with settings appropriate for a release build (e.g.	Scan Verified	Pass	N

	non-debuggable).			
MSTG-CODE-3	Debugging symbols have been removed from native binaries.	Scan Verified	Pass	N
MSTG-CODE-4	Debugging code and developer assistance code (e.g. test code, backdoors, hidden settings) have been removed. The app does not log verbose errors or debugging messages.	Scan Verified	Pass	N
MSTG-CODE-5	All third party components used by the mobile app, such as libraries and frameworks, are identified, and checked for known vulnerabilities.	Self Declare	Pass	N
MSTG-CODE-9	Free security features offered by the toolchain, such as byte-code minification, stack protection, PIE support and automatic reference counting, are activated.	Scan Verified	Pass	N

Self-Declare

MSTG-STORAGE-1

System credential storage facilities need to be used to store sensitive data, such as PII, user credentials or cryptographic keys

Does your app (or library/SDK) use cryptographic keys to encrypt all data that may be considered sensitive, such as PII?

A. Yes
B. **No**

Does your app (or library/SDK) use Android Keystore API to store user credentials?

A. Yes
B. **No**

Does your app (or library/SDK) use Android Keystore API to store cryptographic keys?

A. Yes
B. **No**

MSTG-STORAGE-2

No sensitive data should be stored outside of the app container or system credential storage facilities.

Does your application or library/SDK exclusively store sensitive data within the app container or use system credential storage facilities?

A. **Yes**
B. No

By default, Android SharedPreferences are stored within your app's private internal storage, specifically in the app container (typically at /data/data/<your.package.name>/shared_prefs). This means they aren't stored in a publicly accessible location outside your app's sandbox, ensuring that other apps cannot access them unless the device is rooted or the app explicitly allows external access.

MSTG-STORAGE-12

The app educates the user about the types of personally identifiable information processed, as well as security best practices the user should follow in using the app.

Do you educate users about the types of personally identifiable information (PII) it processes, as well as security best practices they should follow when using the app?

A. **Yes**
B. No

Does your app (or library/SDK) provide a clear and easily accessible link to your privacy policy within the app?

A. **Yes**
B. No

MSTG-CRYPTO-5

The app does not re-use the same cryptographic key for multiple purposes.

Do you have mechanisms in place to audit and verify that each cryptographic key is used exclusively for its designated purpose within the app (or library/SDK) ?

A. **Yes**
B. No

for detected files that get moved into the quarantine, will be encrypted with the same cryptographic key, because for this situation....
Do you verify that each cryptographic key in your app (or library/SDK) is assigned a single, specific purpose to avoid key reuse?

A. **Yes**
B. No

MSTG-AUTH-1

If the app provides users access to a remote service, some form of authentication, such as username/password authentication, is performed at the remote endpoint.

Does your app (or library/SDK) implement authentication at the remote endpoint?

A. **Yes**
B. No
C. N/A (app does not have authentication)

MSTG-AUTH-2

If stateful session management is used, the remote endpoint uses randomly generated session identifiers to authenticate client requests without sending the user`s credentials.

Does your app`s (or library/SDK) remote endpoint use unique and unpredictable session identifiers to authenticate client requests without transmitting the user's credentials?

A. **Yes**
B. No
C. N/A (app does not have authentication)

MSTG-AUTH-3

If stateless token-based authentication is used, the server provides a token that has been signed using a secure algorithm.

If stateless token-based authentication is utilized by your application, does your app server (or library/SDK) provide a token protected using a secure algorithm such as HMAC-SHA256?

A. **Yes**
B. No
C. N/A (app does not have authentication or app does not use stateless token-based authentication)

MSTG-AUTH-4

The remote endpoint terminates the existing session when the user logs out.

Does your app (or library/SDK) terminate the existing session at the remote endpoint when the user logs out?

A. **Yes**
B. No
C. N/A (app does not have authentication)

MSTG-AUTH-5

A password policy exists and is enforced at the remote endpoint.

Does your app (or library/SDK) prevent users from setting passwords they have already used before at the remote endpoint?

A. **Yes**
B. No
C. N/A (app does not have authentication)

Does your app (or library/SDK) enforce a password policy (e.g., minimum length, complexity) on the server/backend side?

A. **Yes**
B. No
C. N/A (app does not have authentication or password is never sent to server (e.g. federated auth or zero-knowledge password proof))

MSTG-AUTH-6

The remote endpoint implements a mechanism to protect against the submission of credentials an excessive number of times.

Does your app (or library/SDK) implement a mechanism at the remote endpoint to protect against the submission of credentials an excessive number of times?

A. **Yes**
B. No
C. N/A (app does not have authentication or password is never sent to server (e.g. federated auth or zero-knowledge password proof))

MSTG-AUTH-7

Sessions are invalidated at the remote endpoint after a predefined period of inactivity and access tokens expire.

Does your app (or library/SDK) implement a mechanism at the remote endpoint (server-side) to automatically invalidate user sessions after a predefined period of inactivity?

A. **Yes**
B. No
C. N/A (app does not have authentication)

Do access tokens used for authentication and authorization within your app (or library/SDK) have a set expiration time, after which the server will reject them as invalid?

A. **Yes**
B. No
C. N/A (app does not have authentication)

MSTG-PLATFORM-4

The app does not export sensitive functionality through IPC facilities, unless these mechanisms are properly protected.

Does your app (or library/SDK) expose sensitive functionality through inter-process communication (IPC) mechanisms?
A. Yes
B. **No**

Does your app (or library/SDK) have security measures like access controls, data validation, and encryption in place to protect sensitive functionality exposed through IPC mechanisms in your app or library/SDK?
A. **Yes**
B. No
C. N/A (app does not have IPC mechanisms)

MSTG-CODE-5

All third party components used by the mobile app, such as libraries and frameworks, are identified, and checked for known vulnerabilities.

Does your app (or library/SDK) use third party libraries?
A. **Yes**
B. No

1. Provide a list of 3P libraries to labs

net.dongliu:apk-parser:2.6.10
com.squareup.okhttp3:okhttp:4.11.0
com.squareup.retrofit2:retrofit:2.9.0
com.squareup.retrofit2:converter-gson:2.9.0
com.zsoltsafrany:needle:1.0.0

com.airbnb.android:lottie:5.2.0
org.greenrobot:eventbus:3.3.1
com.sothree.slidinguppanel:library:3.4.0
com.ogaclejapan.smarttablayout:library:2.0.0@aar
com.futuremind.recyclerfastscroll:fastscroll:0.2.5
com.github.bumptech.glide:glide:4.14.2
commons-codec:commons-codec:1.15
commons-lang:commons-lang:2.6
com.android.volley:volley:1.2.1
com.android.billingclient:billing:7.0.0
com.google.code.gson:gson:2.10.1
com.google.android.play:review:2.0.2
com.google.android.flexbox:flexbox:3.0.0
com.google.android.material:material:1.12.0
com.google.firebase:firebase-messaging:24.1.0
androidx.browser:browser:1.8.0
androidx.cardview:cardview:1.0.0
androidx.work:work-runtime:2.9.1
androidx.appcompat:appcompat:1.7.0
androidx.preference:preference:1.2.1
androidx.recyclerview:recyclerview:1.3.2
androidx.concurrent:concurrent-futures:1.2.0
androidx.swiperefreshlayout:swiperefreshlayout:1.1.0
androidx.lifecycle:lifecycle-extensions:2.2.0
androidx.annotation:annotation:1.8.2

NMI

MSTG-STORAGE-3

No sensitive data is written to application logs.

Finding 1

```
172             try {
173                 return Integer.valueOf(Integer.parseInt(o6));
174             } catch (NumberFormatException unused) {
175                 Log.w("NotificationParams", "Couldn't parse value of " + s(str) + "
176                 return null;
177             }
178         }
179         return null;
180     }
181
182     public JSONArray m(String str) {
183         String o6 = o(str);
184         if (!TextUtils.isEmpty(o6)) {
185             try {
186                 return new JSONArray(o6);
187             } catch (JSONException unused) {
188                 Log.w("NotificationParams", "Malformed JSON for key " + s(str) + ":
189                 return null;
190             }
191         }
192         return null;
193     }
194
195     public String n(Resources resources, String str, String str2) {
196         String[] strArr;
197         String o6 = o(str2);
198         if (!TextUtils.isEmpty(o6)) {
199             return o6;
200         }
201         String o7 = o(str2.concat("_loc_key"));
202         if (TextUtils.isEmpty(o7)) {
203             return null;
204         }
```

Finding 2

```
105                 return null;
106             case 12:
107             case 13:
108             case 14:
109             case 15:
110             case 19:
111             default:
112                 Log.e("GoogleApiAvailability", "Unexpected error code " + i6);
113                 return null;
114             case 16:
115                 Log.e("GoogleApiAvailability", "One of the API components you attem
116                 return null;
117             case 17:
118                 Log.e("GoogleApiAvailability", "The specified account could not be
119                 return e(context, "common_google_play_services_sign_in_failed_title
120             case 20:
121                 Log.e("GoogleApiAvailability", "The current user profile is restric
122                 return e(context, "common_google_play_services_restricted_profile_t
123         }
124     }
125
126     public static String d(Context context, String str, String str2) {
127         Resources resources = context.getResources();
128         String e6 = e(context, str);
129         if (e6 == null) {
130             e6 = resources.getString(R.string.common_google_play_services_unknown_i
131         }
132         return String.format(resources.getConfiguration().locale, e6, str2);
133     }
134
135     public static String e(Context context, String str) {
136         L.f a6;
137         Resources resources;
```

Finding 3

```
179         Y0.b bVar = new Y0.b();
180         this.f3432g = bVar;
181         synchronized (this) {
182             synchronized (bVar) {
183                 bVar.f3345c = this;
184             }
185         }
186         this.f3427b = new D(16);
187         this.f3426a = new p5.g(7);
188         this.f3429d = new b(executorServiceC0365a, executorServiceC0365a2, executor
189         this.f3431f = new a(cVar2);
190         this.f3430e = new v();
191         dVar.f3673d = this;
192     }
193
194     public static void d(String str, long j6, m mVar) {
195         Log.v("Engine", str + " in " + r1.g.a(j6) + "ms, key: " + mVar);
196     }
197
198     public static void g(s sVar) {
199         if (sVar instanceof n) {
200             ((n) sVar).e();
201             return;
202         }
203         throw new IllegalArgumentException("Cannot release anything but an EngineRe
204     }
205
206     public final d a(com.bumptech.glide.g gVar, Object obj, W0.f fVar, int i6, int
207         long j6;
208         if (h) {
209             int i8 = r1.g.f11400b;
210             j6 = SystemClock.elapsedRealtimeNanos();
211         } else {
```

Finding 4

```
91
92     public final boolean d(Object obj) {
93         int i6 = r1.g.f11400b;
94         long elapsedRealtimeNanos = SystemClock.elapsedRealtimeNanos();
95         boolean z5 = false;
96         try {
97             com.bumptech.glide.load.data.e h = this.f3533i.f3362c.a().h(obj);
98             Object a6 = h.a();
99             Object d3 = this.f3533i.d(a6);
100            L1.h hVar = new L1.h(d3, a6, this.f3533i.f3367i);
101            W0.f fVar = this.f3538n.f6461a;
102            g<?> gVar = this.f3533i;
103            e eVar = new e(fVar, gVar.f3372n);
104            InterfaceC0318a a7 = gVar.h.a();
105            a7.b(eVar, hVar);
106            if (Log.isLoggable("SourceGenerator", 2)) {
107                Log.v("SourceGenerator", "Finished encoding source to cache, key: "
108            }
109            if (a7.a(eVar) != null) {
110                this.f3539o = eVar;
111                this.f3536l = new d(Collections.singletonList(this.f3538n.f6461a),
112                this.f3538n.f6463c.b();
113                return true;
114            }
115            if (Log.isLoggable("SourceGenerator", 3)) {
116                Log.d("SourceGenerator", "Attempt to write: " + this.f3539o + ", da
117            }
118            try {
119                this.f3534j.c(this.f3538n.f6461a, h.a(), this.f3538n.f6463c, this.f
120                return false;
121            } catch (Throwable th) {
122                th = th;
123                z5 = true;
```

Finding 5

```
252                     } else {
253                         z5 = false;
254                     }
255                 } catch (IOException | AssertionError unused) {
256                 } catch (Throwable th) {
257                     c6.disconnect();
258                     TrafficStats.clearThreadStatsTag();
259                     throw th;
260                 }
261                 if (z5) {
262                     f6 = C0383c.f(c6);
263                 } else {
264                     C0383c.b(c6, null, str, str2);
265                     if (responseCode != 401 && responseCode != 404) {
266                         if (responseCode != 429) {
267                             if (responseCode < 500 || responseCode >= 600) {
268                                 Log.e("Firebase-Installations", "Firebase Installat
269                                 C0382b.a a7 = c3.f.a();
270                                 a7.f6513c = f.b.BAD_CONFIG;
271                                 f6 = a7.a();
272                             }
273                             c6.disconnect();
274                             TrafficStats.clearThreadStatsTag();
275                         } else {
276                             d.a aVar = d.a.BAD_CONFIG;
277                             throw new d("Firebase servers have received too many re
278                         }
279                     } else {
280                         C0382b.a a8 = c3.f.a();
281                         a8.f6513c = f.b.AUTH_ERROR;
282                         f6 = a8.a();
283                     }
284                 }
```

Finding 6

```
21
22      /* renamed from: k  reason: collision with root package name */
23      public final long f3670k = 262144000;
24
25      /* renamed from: i  reason: collision with root package name */
26      public final f f3668i = new f();
27
28      @Deprecated
29      public C0320c(File file) {
30          this.f3669j = file;
31      }
32
33      @Override // a1.InterfaceC0318a
34      public final File a(W0.f fVar) {
35          String b6 = this.f3668i.b(fVar);
36          if (Log.isLoggable("DiskLruCacheWrapper", 2)) {
37              Log.v("DiskLruCacheWrapper", "Get: Obtained: " + b6 + " for for Key: " +
38          }
39          try {
40              a.e o6 = c().o(b6);
41              if (o6 == null) {
42                  return null;
43              }
44              return o6.f3064a[0];
45          } catch (IOException e6) {
46              if (!Log.isLoggable("DiskLruCacheWrapper", 5)) {
47                  return null;
48              }
49              Log.w("DiskLruCacheWrapper", "Unable to get from disk cache", e6);
50              return null;
51          }
52      }
53
```

Finding 7

```
690         return this.f4867x;
691     }
692
693     @Override // android.view.View
694     public int getSuggestedMinimumHeight() {
695         return Math.max(super.getSuggestedMinimumHeight(), getPaddingBottom() + get
696     }
697
698     @Override // android.view.View
699     public int getSuggestedMinimumWidth() {
700         return Math.max(super.getSuggestedMinimumWidth(), getPaddingRight() + getPa
701     }
702
703     public final int h(int i6) {
704         int[] iArr = this.f4860q;
705         if (iArr == null) {
706             Log.e("CoordinatorLayout", "No keylines defined for " + this + " - atte
707             return 0;
708         } else if (i6 >= 0 && i6 < iArr.length) {
709             return iArr[i6];
710         } else {
711             Log.e("CoordinatorLayout", "Keyline index " + i6 + " out of range for "
712             return 0;
713         }
714     }
715
716     @Override // P.InterfaceC0307o
717     public final void i(View view, int i6) {
718         C0309q c0309q = this.f4851A;
719         if (i6 == 1) {
720             c0309q.f2634b = 0;
721         } else {
722             c0309q.f2633a = 0;
```

Finding 8

```
184                     if (string == null) {
185                         c6 = null;
186                     } else {
187                         c6 = d3.f5291c.c(string);
188                         if (c6 == null) {
189                             d3.c0(new IllegalStateException("Fragment no longer exi
190                             throw null;
191                         }
192                     }
193                     if (c6 != null) {
194                         while (arrayList2.size() <= parseInt) {
195                             arrayList2.add(null);
196                         }
197                         c6.U(false);
198                         arrayList2.set(parseInt, c6);
199                     } else {
200                         Log.w("FragmentStatePagerAdapt", "Bad fragment at key ".con
201                     }
202                 }
203             }
204         }
205     }
206
207     @Override // B0.a
208     public final Parcelable i() {
209         Bundle bundle;
210         ArrayList<ComponentCallbacksC0339j.f> arrayList = this.f5076f;
211         if (arrayList.size() > 0) {
212             bundle = new Bundle();
213             ComponentCallbacksC0339j.f[] fVarArr = new ComponentCallbacksC0339j.f[a
214             arrayList.toArray(fVarArr);
215             bundle.putParcelableArray("states", fVarArr);
216         } else {
```

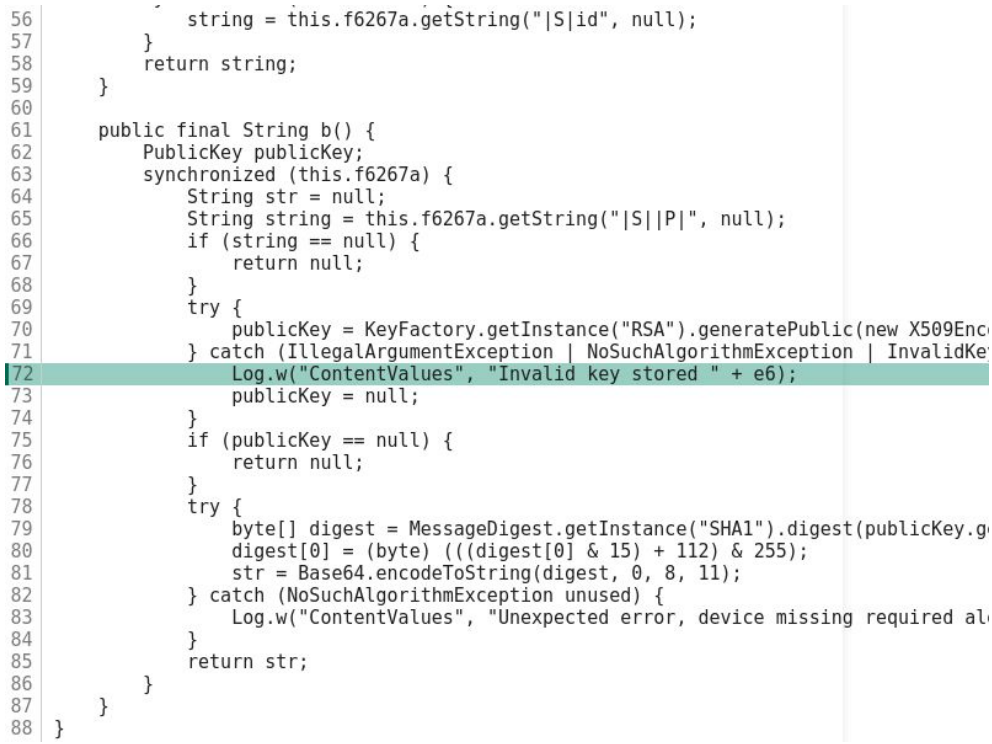
Finding 9

```
11  import java.util.ArrayList;
12  import u.i;
13  /* loaded from: classes.dex */
14  public abstract class PreferenceGroup extends Preference {
15
16      /* renamed from: H  reason: collision with root package name */
17      public final ArrayList f5498H;
18
19      public PreferenceGroup(Context context, AttributeSet attributeSet, int i6) {
20          super(context, attributeSet, i6);
21          new i();
22          new Handler(Looper.getMainLooper());
23          this.f5498H = new ArrayList();
24          TypedArray obtainStyledAttributes = context.obtainStyledAttributes(attribute
25          obtainStyledAttributes.getBoolean(2, obtainStyledAttributes.getBoolean(2, tr
26          if (obtainStyledAttributes.hasValue(1) && obtainStyledAttributes.getInt(1, o
27              Log.e("PreferenceGroup", getClass().getSimpleName().concat(" should have
28          }
29          obtainStyledAttributes.recycle();
30      }
31
32      @Override // androidx.preference.Preference
33      public final void f(boolean z5) {
34          super.f(z5);
35          int size = this.f5498H.size();
36          for (int i6 = 0; i6 < size; i6++) {
37              Preference preference = (Preference) this.f5498H.get(i6);
38              if (preference.f5470u == z5) {
39                  preference.f5470u = !z5;
40                  preference.f(preference.l());
41                  preference.c();
42              }
43          }
```

Finding 10

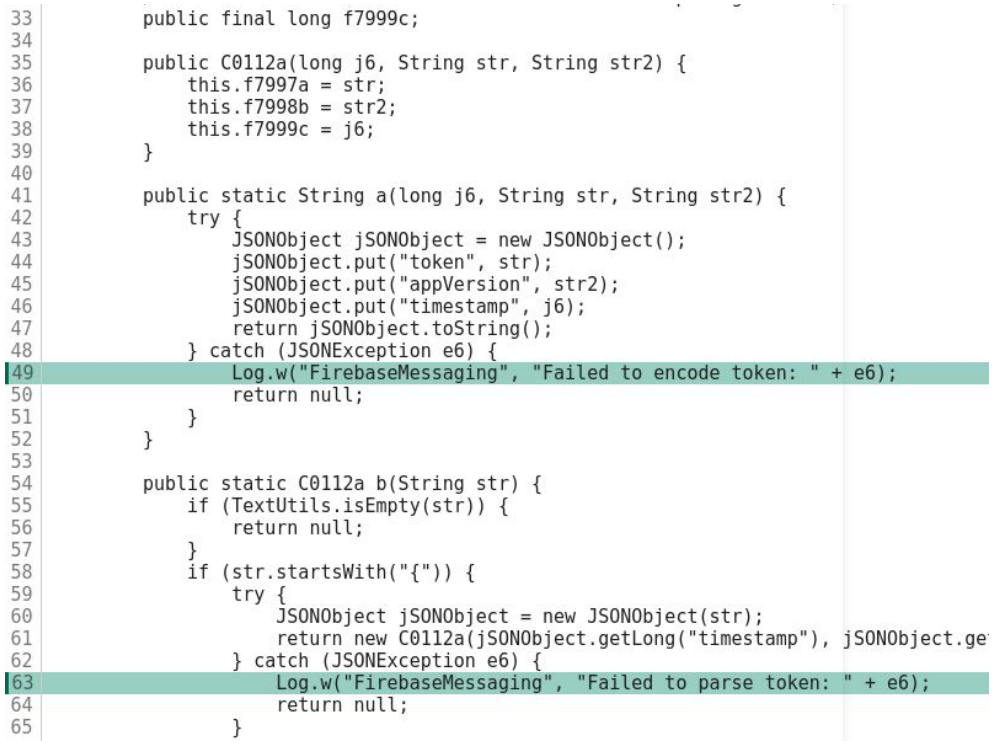
```
146                 if (d3 == null) {
147                     deviceState = null;
148                 } else {
149                     deviceState = d3.getDeviceState();
150                 }
151                 if (deviceState == null) {
152                     deviceState = new SidecarDeviceState();
153                 }
154                 u e6 = jVar.e(sidecarWindowLayoutInfo, deviceState);
155                 b bVar = this.f6065a.f6059e;
156                 if (bVar != null) {
157                     bVar.a(activity, e6);
158                     return;
159                 }
160                 return;
161             }
162             Log.w("SidecarCompat", "Unable to resolve activity from window token. M
163         }
164     }
165
166     /* loaded from: classes.dex */
167     public static final class a {
168         public static IBinder a(Activity activity) {
169             Window window;
170             WindowManager.LayoutParams attributes;
171             if (activity == null || (window = activity.getWindow()) == null || (att
172                 return null;
173             }
174             return attributes.token;
175         }
176
177         public static SidecarInterface b(Context context) {
178             A4.i.f(context, "context");
```

Finding 11



```
56              string = this.f6267a.getString("|S|id", null);
57          }
58          return string;
59      }
60
61      public final String b() {
62          PublicKey publicKey;
63          synchronized (this.f6267a) {
64              String str = null;
65              String string = this.f6267a.getString("|S||P|", null);
66              if (string == null) {
67                  return null;
68              }
69              try {
70                  publicKey = KeyFactory.getInstance("RSA").generatePublic(new X509Enc
71              } catch (IllegalArgumentException | NoSuchAlgorithmException | InvalidKe
72                  Log.w("ContentValues", "Invalid key stored " + e6);
73                  publicKey = null;
74              }
75              if (publicKey == null) {
76                  return null;
77              }
78              try {
79                  byte[] digest = MessageDigest.getInstance("SHA1").digest(publicKey.g
80                  digest[0] = (byte) (((digest[0] & 15) + 112) & 255);
81                  str = Base64.encodeToString(digest, 0, 8, 11);
82              } catch (NoSuchAlgorithmException unused) {
83                  Log.w("ContentValues", "Unexpected error, device missing required al
84              }
85              return str;
86          }
87      }
88  }
```

Finding 12



```
33      public final long f7999c;
34
35      public C0112a(long j6, String str, String str2) {
36          this.f7997a = str;
37          this.f7998b = str2;
38          this.f7999c = j6;
39      }
40
41      public static String a(long j6, String str, String str2) {
42          try {
43              JSONObject jSONObject = new JSONObject();
44              jSONObject.put("token", str);
45              jSONObject.put("appVersion", str2);
46              jSONObject.put("timestamp", j6);
47              return jSONObject.toString();
48          } catch (JSONException e6) {
49              Log.w("FirebaseMessaging", "Failed to encode token: " + e6);
50              return null;
51          }
52      }
53
54      public static C0112a b(String str) {
55          if (TextUtils.isEmpty(str)) {
56              return null;
57          }
58          if (str.startsWith("{")) {
59              try {
60                  JSONObject jSONObject = new JSONObject(str);
61                  return new C0112a(jSONObject.getLong("timestamp"), jSONObject.ge
62              } catch (JSONException e6) {
63                  Log.w("FirebaseMessaging", "Failed to parse token: " + e6);
64                  return null;
65              }
```

Finding 13

```
73          NetworkInfo networkInfo;
74          ConnectivityManager connectivityManager = (ConnectivityManager) this.f9509k
75          if (connectivityManager != null) {
76              networkInfo = connectivityManager.getActiveNetworkInfo();
77          } else {
78              networkInfo = null;
79          }
80          if (networkInfo != null && networkInfo.isConnected()) {
81              return true;
82          }
83          return false;
84      }
85
86      public final boolean b() {
87          try {
88              if (this.f9509k.a() == null) {
89                  Log.e("FirebaseMessaging", "Token retrieval failed: null");
90                  return false;
91              } else if (Log.isLoggable("FirebaseMessaging", 3)) {
92                  Log.d("FirebaseMessaging", "Token successfully retrieved");
93                  return true;
94              } else {
95                  return true;
96              }
97          } catch (IOException e6) {
98              String message = e6.getMessage();
99              if (!"SERVICE_NOT_AVAILABLE".equals(message) && !"INTERNAL_SERVER_ERROR
100                 if (e6.getMessage() == null) {
101                     Log.w("FirebaseMessaging", "Token retrieval failed without exce
102                     return false;
103                 }
104                 throw e6;
105             }
```

Finding 14

```
81          bundle.putString("osv", Integer.toString(Build.VERSION.SDK_INT));
82          bundle.putString("app_ver", this.f9600b.a());
83          bundle.putString("app_ver_name", this.f9600b.b());
84          K2.e eVar2 = this.f9599a;
85          eVar2.a();
86          try {
87              str3 = Base64.encodeToString(MessageDigest.getInstance("SHA-1").digest(
88          } catch (NoSuchAlgorithmException unused) {
89              str3 = "[HASH-ERROR]";
90          }
91          bundle.putString("firebase-app-name-hash", str3);
92          try {
93              String a6 = ((Z2.g) e2.l.a(this.f9604f.a())).a();
94              if (!TextUtils.isEmpty(a6)) {
95                  bundle.putString("Goog-Firebase-Installations-Auth", a6);
96              } else {
97                  Log.w("FirebaseMessaging", "FIS auth token is empty");
98              }
99          } catch (InterruptedException e6) {
100             e = e6;
101             Log.e("FirebaseMessaging", "Failed to get FIS auth token", e);
102             bundle.putString("appid", (String) e2.l.a(this.f9604f.b()));
103             bundle.putString("cliv", "fcm-24.1.0");
104             gVar = this.f9603e.get();
105             InterfaceC0582f interfaceC0582f = this.f9602d.get();
106             if (gVar != null) {
107                 return;
108             }
109             return;
110         } catch (ExecutionException e7) {
111             e = e7;
112             Log.e("FirebaseMessaging", "Failed to get FIS auth token", e);
113             bundle.putString("appid", (String) e2.l.a(this.f9604f.b()));
```

We would like to clarify the following points:

1. **No Sensitive User Data Logged**
   Our application does not record user credentials, personally identifiable information (PII), or license keys in any logs. Any potentially sensitive text input by the user is handled internally and never written to device logs.
2. **Referenced Classes from Third-Party / Google Libraries**
   The files mentioned (e.g., CoordinatorLayout.java, PreferenceGroup.java, various Glide classes) originate from standard Android libraries or open-source components like Glide . We have not modified these libraries to log private or sensitive data. Glide manages image loading and caching, and does not record user-input text or confidential information in logs.
3. **Compliance with MSTG-STORAGE-3**
   We ensure that no confidential or user-sensitive data is exposed through any logging mechanism. Our production logs are strictly limited to operational or debugging messages unrelated to personal or license data.

MSTG-STORAGE-7

No sensitive data, such as passwords or pins, is exposed through the user interface.

Finding 1

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:orie
3      <RelativeLayout android:orientation="vertical" android:background="@drawable/myps
4          <View android:background="@drawable/myps_slider_indicator" android:layout_wi
5          <TextView android:textSize="17sp" android:textColor="@color/colorTint" andro
6      </RelativeLayout>
7      <include android:id="@+id/divider" layout="@layout/fragment_border"/>
8      <LinearLayout android:orientation="vertical" android:background="@color/colorSli
9          <EditText android:textSize="14sp" android:textColor="@color/colorTint" andro
10         <LinearLayout android:gravity="center" android:orientation="horizontal" andr
11             <TextView android:textSize="15sp" android:textColor="@color/colorTint" a
12         </LinearLayout>
13     </LinearLayout>
14 </LinearLayout>
15
```

Finding 2

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:orie
    <RelativeLayout android:orientation="vertical" android:background="@drawable/myps
        <View android:background="@drawable/myps_slider_indicator" android:layout_wi
        <TextView android:textSize="17sp" android:textColor="@color/colorTint" andro
    </RelativeLayout>
    <include android:id="@+id/divider" layout="@layout/fragment_border"/>
    <LinearLayout android:orientation="vertical" android:background="@color/colorSli
        <EditText android:textSize="14sp" android:textColor="@color/colorTint" andro
        <LinearLayout android:gravity="center" android:orientation="horizontal" andr
            <TextView android:textSize="15sp" android:textColor="@color/colorTint" a
        </LinearLayout>
    </LinearLayout>
</LinearLayout>

```

We would like to clarify:

1. **License Keys vs. Sensitive User Data**
   - The field in myps_activity_activate_slider.xml is for entering a **license key** , not a password, PIN, or personally identifiable information (PII).
   - It does **not** grant access to personal user data or user accounts, so it is not considered confidential in the same sense as passwords or PINs.
2. **No Exposure of Confidential Information**
   - We do not display any user credentials, session tokens, or other data that could compromise user security.
   - The EditText is solely for product activation, and its content reveals no sensitive user details.
3. **Compliance with MSTG-STORAGE-7**
   - As the license key is not classified as sensitive user data (like a password or PIN), displaying it in the UI does **not** violate MSTG-STORAGE-7.
   - No other UI components expose private or sensitive data.

Scan Verified

In this section you can find all the results based on the automatic evaluation, for the test cases in PASS it can be seen as informative, for the results marked as Fail you need to fix the issues or provide some feedback or justification in the respective area.

FAIL

PASS

MSTG-STORAGE-5

The keyboard cache is disabled on text inputs that process sensitive data.

After further analysis the application seems to comply successfully with the requirement.

MSTG-CRYPTO-1

The app does not rely on symmetric cryptography with hardcoded keys as a sole method of encryption.

Finding 1

```
3  import E0.t;
4  import G3.a;
5  import H3.a;
6  import J3.c;
7  import N3.d;
8  import N3.h;
9  import N3.i;
10 import android.content.Context;
11 import android.os.ParcelFileDescriptor;
12 import com.protectstar.antispy.DeviceStatus;
13 import java.io.File;
14 import java.io.FileInputStream;
15 import java.io.FileOutputStream;
16 import java.io.IOException;
17 import java.nio.ByteBuffer;
18 import java.nio.channels.FileChannel;
19 import java.security.SecureRandom;
20 import java.util.ArrayList;
21 import java.util.Arrays;
22 import java.util.HashMap;
23 import java.util.HashSet;
24 import java.util.Iterator;
25 import java.util.LinkedHashSet;
26 import java.util.Objects;
27 import java.util.UUID;
28 import javax.crypto.Cipher;
29 import javax.crypto.CipherInputStream;
30 import javax.crypto.CipherOutputStream;
31 import javax.crypto.spec.IvParameterSpec;
32 import javax.crypto.spec.SecretKeySpec;
33 import k3.InterfaceC0612b;
34 /* loaded from: classes.dex */
35 public final class b {
```

Finding 2

```
1  package N3;
2
3  import J3.c;
4  import T3.m;
5  import android.os.Build;
6  import java.io.BufferedInputStream;
7  import java.io.File;
8  import java.io.FileInputStream;
9  import java.io.FileNotFoundException;
10 import java.io.IOException;
11 import java.security.MessageDigest;
12 import java.security.NoSuchAlgorithmException;
13 import java.util.ArrayList;
14 import java.util.HashMap;
15 import java.util.HashSet;
16 import java.util.Iterator;
17 import java.util.LinkedHashMap;
18 import java.util.Map;
19 import java.util.Objects;
20 import java.util.Set;
21 /* loaded from: classes.dex */
22 public final class h {
23
24     /* renamed from: a  reason: collision with root package name */
25     public static final char[] f2244a = "0123456789abcdef".toCharArray();
26
27     /* JADX WARN: Type inference failed for: r0v4, types: [java.lang.Object, N3.g]
28     public static void a(HashMap<String, HashSet<e>> hashMap, String str, e eVar) {
29         Object computeIfAbsent;
30         if (str != null && !str.isEmpty()) {
31             if (Build.VERSION.SDK_INT >= 24) {
32                 computeIfAbsent = hashMap.computeIfAbsent(str, new Object());
33                 ((HashSet) computeIfAbsent).add(eVar);
```

We would like to clarify the following:

1. **java_source/H3/b.java (AES-256 Key & IV for Quarantine)**
   - The hardcoded AES key and IV in this file are **not** used to protect sensitive user data. Instead, we use them to **encrypt malicious or suspicious files** when quarantining, rendering those files temporarily unusable.
   - This mechanism prevents the file from being directly executed or accessed while in quarantine. The key is merely a **technical measure** to neutralize the threat, but it is **not** intended for protecting personally identifiable information (PII).
2. **No Sole Reliance on Hardcoded Keys**
   - Because these files are malicious, the encryption key does not secure user-owned data and **does not** expose any user credentials, secrets, or personal data.
   - If the user chooses to restore the quarantined file, the same key allows reversion. This process does **not** grant access to confidential user information, as none is stored this way.
3. **java_source/N3/h.java (Helper Methods & SecureRandom)**
   - The methods flagged here deal with **securely overwriting files** using either random data from SecureRandom or a fixed pattern (e.g., 0xFF).
   - These methods are employed for secure deletion workflows, **not** for encrypting sensitive user data with a hardcoded key.
4. **Compliance with MSTG-CRYPTO-1**

- We confirm that our app does **not** rely on a hardcoded symmetric key for protecting user data.
- The hardcoded key you identified is simply part of our quarantine functionality, ensuring malicious files are rendered inert. All actual user credentials or personal data, if any, would be secured through **proper cryptographic measures** (e.g., using SecureRandom or other recommended approaches).

After further analysis the application seems to comply successfully with the requirement.

MSTG-CRYPTO-2

The app uses proven implementations of cryptographic primitives.

After further analysis the application seems to comply successfully with the requirement.

MSTG-CRYPTO-3

The app uses cryptographic primitives that are appropriate for the particular use-case, configured with parameters that adhere to industry best practices.

Finding 1

```
111     private HashSet<a.C0012a> f1229p = new HashSet<>();
112
113     /* loaded from: classes.dex */
114     public enum a {
115         SUCCESS,
116         GENERAL,
117         MISSING_ROOT,
118         WRITE
119     }
120
121     public b(File file) {
122         this.f1215a = file.getName();
123         this.f1217c = file.getAbsolutePath();
124     }
125
126     public static void h(FileInputStream fileInputStream, FileOutputStream fileOutp
127         Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
128         cipher.init(2, new SecretKeySpec(f1214t, "AES"), new IvParameterSpec(f1213s
129         CipherInputStream cipherInputStream = new CipherInputStream(fileInputStream
130         try {
131             byte[] bArr = new byte[8192];
132             while (true) {
133                 int read = cipherInputStream.read(bArr);
134                 if (read != -1) {
135                     fileOutputStream.write(bArr, 0, read);
136                 } else {
137                     fileOutputStream.flush();
138                     cipherInputStream.close();
139                     return;
140                 }
141             }
142         } catch (Throwable th) {
143             try {
```

Finding 2

```
168             int ceil = (int) Math.ceil(length / min);
169             for (int i7 = 0; i7 < ceil; i7++) {
170                 if (i7 > 0 && i7 == ceil - 1) {
171                     i6 = (int) (length - (min * i7));
172                 } else {
173                     i6 = min;
174                 }
175                 byte[] bArr = new byte[i6];
176                 randomAccessFile.seek(min * i7);
177                 int i8 = 0;
178                 while (i8 < i6) {
179                     i8 += randomAccessFile.read(bArr, i8, i6 - i8);
180                 }
181                 if (i8 == i6) {
182                     FileOutputStream fileOutputStream = (FileOutputStream) tVar.f73
183                     try {
184                         Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
185                         cipher.init(2, new SecretKeySpec(H3.b.f1214t, "AES"), new I
186                         fileOutputStream.write(cipher.doFinal(bArr));
187                     } catch (Exception e6) {
188                         e6.printStackTrace();
189                     }
190                 } else {
191                     throw new IOException("Unexpected read size. current: " + i8 +
192                 }
193             }
194             randomAccessFile.close();
195         } catch (FileNotFoundException unused) {
196         } catch (Throwable th) {
197             th.printStackTrace();
198         }
199     }
200 }
```

We would like to clarify the following:

1. **java_source/H3/b.java (Encrypt/Decrypt for Quarantine)**
   - These methods encrypt and decrypt **malicious files** when moving them into or out of our quarantine.
   - Since the quarantined files are **not** user-sensitive (e.g., not personal documents or credentials), this encryption primarily renders malicious content unusable until (and unless) it is intentionally restored.
   - We **do** use a modern cryptographic algorithm (AES-256) to perform byte-level modifications. This choice ensures that once encrypted, the file is effectively neutralized in storage.
2. **No User-Sensitive Data**
   - The quarantine mechanism is designed for potentially harmful files, rather than for protecting user secrets or personal data.
   - As a result, while we apply AES-based encryption, it is not being used for **user credential** encryption or similarly sensitive user information.
3. **java_source/N3/d.java (File Handling Helpers)**
   - This file primarily contains generic file I/O operations (copying files, writing/reading bytes, etc.).
   - It does **not** introduce additional cryptographic functionality that would affect sensitive user data or store cryptographic keys.
4. **Industry Best Practices**
   - We follow recommended guidelines for the cryptographic primitives in use, leveraging AES-256 from standard libraries, and avoiding outdated or insecure algorithms.

- Our code does not rely on deprecated modes such as ECB, and we remain mindful of using strong ciphers even though the data here is **not** user-sensitive.

In conclusion, our app's cryptographic usage aligns with MSTG-CRYPTO-3 for its specific purpose (quarantine encryption), and **no** sensitive user data is exposed or at risk.
After further analysis the application seems to comply successfully with the requirement.

MSTG-CRYPTO-4

The app does not use cryptographic protocols or algorithms that are widely considered deprecated for security purposes.

After further analysis the application seems to comply successfully with the requirement.

MSTG-CRYPTO-6

All random values are generated using a sufficiently secure random number generator.

After further analysis the application seems to comply successfully with the requirement.

MSTG-NETWORK-1

Data is encrypted on the network using TLS. The secure channel is used consistently throughout the app.

After further analysis the application seems to comply successfully with the requirement.

MSTG-NETWORK-2

The TLS settings are in line with current best practices, or as close as possible if the mobile operating system does not support the recommended standards.

After further analysis the application seems to comply successfully with the requirement.

MSTG-NETWORK-3

The app verifies the X.509 certificate of the remote endpoint when the secure channel is established.

After further analysis the application seems to comply successfully with the requirement.

MSTG-PLATFORM-1

The app only requests the minimum set of permissions necessary.

After further analysis the application seems to comply successfully with the requirement.

MSTG-PLATFORM-2

All inputs from external sources and the user are validated and if necessary sanitized. This includes data received via the UI, IPC mechanisms such as intents, custom URLs, and network sources.

Finding 1

```
23          this.f768i = j6;
24      }
25
26      @Override // F1.l.a
27      public Object apply(Object obj) {
28          SQLiteDatabase sQLiteDatabase = (SQLiteDatabase) obj;
29          c.a aVar = (c.a) this.f770k;
30          String num = Integer.toString(aVar.getNumber());
31          String str = (String) this.f769j;
32          boolean booleanValue = ((Boolean) F1.l.B(sQLiteDatabase.rawQuery("SELECT 1 FI
33          long j6 = this.f768i;
34          if (!booleanValue) {
35              ContentValues contentValues = new ContentValues();
36              contentValues.put("log_source", str);
37              contentValues.put("reason", Integer.valueOf(aVar.getNumber()));
38              contentValues.put("events_dropped_count", Long.valueOf(j6));
39              sQLiteDatabase.insert("log_event_dropped", null, contentValues);
40          } else {
41              sQLiteDatabase.execSQL("UPDATE log_event_dropped SET events_dropped_coun
42          }
43          return null;
44      }
45
46      @Override // G1.b.a
47      public Object b() {
48          o oVar = (o) this.f769j;
49          oVar.f773c.D(oVar.f777g.b() + this.f768i, (y1.j) this.f770k);
50          return null;
51      }
52  }
53
```

Finding 2

```
124                 String str = jVar2.f12605a;
125                 if (sQLiteDatabase.update("transport_contexts", contentValues, "bac
126                     contentValues.put("backend_name", str);
127                     contentValues.put("priority", Integer.valueOf(I1.a.a(eVar)));
128                     sQLiteDatabase.insert("transport_contexts", null, contentValues
129                 }
130                 return null;
131             }
132         });
133     }
134
135     @Override // F1.d
136     public final void H(Iterable<h> iterable) {
137         if (!iterable.iterator().hasNext()) {
138             return;
139         }
140         String str = "UPDATE events SET num_attempts = num_attempts + 1 WHERE _id i
141         SQLiteDatabase k6 = k();
142         k6.beginTransaction();
143         try {
144             k6.compileStatement(str).execute();
145             Cursor rawQuery = k6.rawQuery("SELECT COUNT(*), transport_name FROM eve
146             while (rawQuery.moveToNext()) {
147                 g(rawQuery.getInt(0), c.a.MAX_RETRIES_REACHED, rawQuery.getString(1
148             }
149             rawQuery.close();
150             k6.compileStatement("DELETE FROM events WHERE num_attempts >= 16").exec
151             k6.setTransactionSuccessful();
152         } finally {
153             k6.endTransaction();
154         }
155     }
156
```

Finding 3

```
440             } else {
441                 hVar.add(str);
442             }
443         }
444         C0726h.b(hVar);
445         Object[] array = hVar.toArray(new String[0]);
446         A4.i.d(array, "null cannot be cast to non-null type kotlin.Array<T of kotli
447         return (String[]) array;
448     }
449
450     public final void d(InterfaceC0757b interfaceC0757b, int i6) {
451         String str;
452         interfaceC0757b.s("INSERT OR IGNORE INTO room_table_modification_log VALUES
453         String str2 = this.f10509e[i6];
454         String[] strArr = f10504o;
455         for (int i7 = 0; i7 < 3; i7++) {
456             String str3 = "CREATE TEMP TRIGGER IF NOT EXISTS " + a.a(str2, str) + "
457             A4.i.e(str3, "StringBuilder().apply(builderAction).toString()");
458             interfaceC0757b.s(str3);
459         }
460     }
461
462     public final void e(InterfaceC0757b interfaceC0757b) {
463         A4.i.f(interfaceC0757b, "database");
464         if (interfaceC0757b.Y()) {
465             return;
466         }
467         try {
468             ReentrantReadWriteLock.ReadLock readLock = this.f10505a.f10536i.readLoc
469             A4.i.e(readLock, "readWriteLock.readLock()");
470             readLock.lock();
471             try {
472                 synchronized (this.f10515l) {
```

We would like to clarify the following:

1. **Google-Provided Library Code**
   The files you referenced (E1/n.java, F1/l.java, m0/C0644i.java) appear to be from a Google-provided, open-source library. We have not modified or customized these library files.
2. **Input Validation in Our Application**
   Although these classes belong to a standard library, **our own code** ensures that any user-supplied or external data is validated and sanitized before being passed to these library methods. We do not allow untrusted or potentially malicious inputs to reach sensitive routines.
3. **Adherence to MSTG-PLATFORM-2**
   We take a **defensive approach** , validating all incoming data—whether from UI inputs, intents, URLs, or network sources—to mitigate the risk of injection attacks or security breaches. Our integration of Google's library does not bypass any of these safeguards.
4. **Regular Updates and Security Monitoring**
   We keep our dependencies (including Google libraries) up to date, ensuring we benefit from the latest security patches and adhere to current best practices.

After further analysis, we confirm that our application **fully complies** with MSTG-PLATFORM-2.
After further analysis the application seems to comply successfully with the requirement.

MSTG-PLATFORM-3

The app does not export sensitive functionality via custom URL schemes, unless these mechanisms are properly protected.

After further analysis the application seems to comply successfully with the requirement.

MSTG-CODE-1

The app is signed and provisioned with a valid certificate, of which the private key is properly protected.

After further analysis the application seems to comply successfully with the requirement.

MSTG-CODE-2

The app has been built in release mode, with settings appropriate for a release build (e.g. non-debuggable).

After further analysis the application seems to comply successfully with the requirement.

MSTG-CODE-3

Debugging symbols have been removed from native binaries.

After further analysis the application seems to comply successfully with the requirement.

MSTG-CODE-4

Debugging code and developer assistance code (e.g. test code, backdoors, hidden settings) have been removed. The app does not log verbose errors or debugging messages.

Finding 1

```
12 public final /* synthetic */ class e implements Y2.a {
13
14     /* renamed from: a  reason: collision with root package name */
15     public final /* synthetic */ int f2188a;
16
17     public /* synthetic */ e(int i6) {
18         this.f2188a = i6;
19     }
20
21     @Override // Y2.a
22     public final Object get() {
23         switch (this.f2188a) {
24             case ReviewErrorCode.NO_ERROR /* 0 */:
25                 return Collections.emptySet();
26             case 1:
27                 o<ScheduledExecutorService> oVar = ExecutorsRegistrar.f7967a;
28                 StrictMode.ThreadPolicy.Builder detectNetwork = new StrictMode.Threa
29                 int i6 = Build.VERSION.SDK_INT;
30                 if (i6 >= 23) {
31                     detectNetwork.detectResourceMismatches();
32                     if (i6 >= 26) {
33                         detectNetwork.detectUnbufferedIo();
34                     }
35                 }
36                 return new O2.h(Executors.newFixedThreadPool(4, new O2.a("Firebase B
37             case 2:
38                 o<ScheduledExecutorService> oVar2 = ExecutorsRegistrar.f7967a;
39                 return Executors.newSingleThreadScheduledExecutor(new O2.a("Firebase
40             default:
41                 com.google.firebase.messaging.a aVar = FirebaseMessaging.f7976m;
42                 return null;
43         }
44     }
```

Finding 2

```
1  package O2;
2
3  import N2.o;
4  import android.os.StrictMode;
5  import com.google.firebase.concurrent.ExecutorsRegistrar;
6  import java.util.concurrent.Executors;
7  import java.util.concurrent.ScheduledExecutorService;
8  /* loaded from: classes.dex */
9  public final /* synthetic */ class k implements Y2.a {
10     @Override // Y2.a
11     public final Object get() {
12         o<ScheduledExecutorService> oVar = ExecutorsRegistrar.f7967a;
13         return new h(Executors.newFixedThreadPool(Math.max(2, Runtime.getRuntime().a
14     }
15 }
16
```

Finding 3

```
66
67         /* renamed from: b1.a$b$a  reason: collision with other inner class name */
68         /* loaded from: classes.dex */
69         public class RunnableC0083a implements Runnable {
70
71             /* renamed from: i  reason: collision with root package name */
72             public final /* synthetic */ Runnable f6249i;
73
74             public RunnableC0083a(Runnable runnable) {
75                 this.f6249i = runnable;
76             }
77
78             @Override // java.lang.Runnable
79             public final void run() {
80                 b bVar = b.this;
81                 if (bVar.f6247d) {
82                     StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder(
83                 }
84                 try {
85                     this.f6249i.run();
86                 } catch (Throwable th) {
87                     bVar.f6246c.getClass();
88                     if (Log.isLoggable("GlideExecutor", 6)) {
89                         Log.e("GlideExecutor", "Request threw uncaught throwable", t
90                     }
91                 }
92             }
93         }
94
95         public b(ThreadFactoryC0081a threadFactoryC0081a, String str, boolean z5) {
96             c.C0084a c0084a = c.f6251a;
97             this.f6248e = new AtomicInteger();
98             this.f6244a = threadFactoryC0081a;
```

We would like to clarify the following:

1. **Referenced Library (Glide)**
   The files you have noted (N2/e.java, O2/k.java, b1/ExecutorServiceC0365a.java) appear to stem from the open-source Glide library. Glide is utilized for efficient image loading and caching, ensuring smooth scrolling and optimized performance.
2. **StrictMode Policy Check (No Debug Backdoor)**
   Portions of the flagged code involve **StrictMode** , which is used to detect inadvertent network calls on the main (UI) thread. This is a standard best practice to prevent performance bottlenecks (ANRs), not a debugging or backdoor mechanism. We do not introduce or ship verbose logs, hidden test code, or developer assistance backdoors in our production build.
3. **No Verbose Logging or Debugging Routines**
   We confirm that we do **not** log sensitive information or carry additional debugging instructions in our release version. The portions of Glide included in our app do not emit sensitive data; they simply support stable performance and efficient caching.

After reviewing these details, we confirm that our application does not contain any debug code, backdoors, or hidden settings. We therefore **comply** with MSTG-CODE-4. If you require further information, please let us know.
After further analysis the application seems to comply successfully with the requirement.

MSTG-CODE-9

Free security features offered by the toolchain, such as byte-code minification, stack protection, PIE support and automatic reference counting, are activated.

After further analysis the application seems to comply successfully with the requirement.