ANDROID RANSOMWARE: TURNING CRYPTOLOCKER INTO CRYPTO UNLOCKER

Alexander Adamov
NioGuard Security Lab, Ukraine

Email: ada@nioguard.com

ABSTRACT

These days, we see an increasing number of new pieces of ransomware for *Android* devices. They adopt new social engineering, communication and encryption techniques such as the use of Tor and advanced encryption algorithms (RSA-1024 and even elliptic curve cryptography). However, the majority of *Android* cryptolockers are simple enough to be disassembled and reused to restore encrypted data. Simple reverse engineering techniques can be used to disassemble and patch the cryptolocker. As a result, it can be turned into a crypto *unlocker* to decrypt encrypted the user's files on an SD card.

ANDROID RANSOMWARE OVERVIEW

Android ransomware has become popular among malware writers who use social engineering techniques and suspicious app stores to publish their applications in order to get users infected with ransomware. While in *Windows* a huge variety of blockers have been doing the rounds for the last decade, in *Android* they have come into play only during the last few years.

Let us take a look at the evolution of this threat, how it has been growing over the last several years, and whether there are any samples we should be worried by so far.

The first sample of *Android* ransomware to be discovered was dated 2012 and called Reveton FBI or Police Locker. The malware shows a fake police warning that asks the victim to pay $200 within 48 hours for copyright offences [1].

Another example of police ransomware comes from 2014: Koler was supposedly created by the same team as Reveton. It works using the same scheme – demanding that the victim pays a ransom of $100–$300 via *Ukash* and *PaysafeCard* in order to unlock the infected device. In addition, the Koler campaign has a comprehensive infrastructure to distribute itself through a specially crafted network of porn sites made with the help of the WebLoader service. Specially for *Windows* users, Koler was empowered with the Angler exploit kit that targets vulnerabilities in Silverlight, Adobe Flash, and Java in case a victim is surfing from *Internet Explorer* [2].

FakeDefender was discovered in 2013 and described by researchers from *Symantec* [3] and *Fortinet* [4]. This is a simple example of the FakeAV class of malware that is well known among *Windows* users. The same trick works well on the *Android* platform: the victim is urged to buy a fake anti-malware app in order to remove malicious programs that have allegedly been found on the device. However, payment does not fix the supposed problem, and even worse, it leads to a leakage of the victim's credit card information.

Finally, the class of cryptolockers is represented by the widespread SimpleLocker family that encrypts data on an SD card and demands a ransom in order to get them back. We will analyse this in the next section.

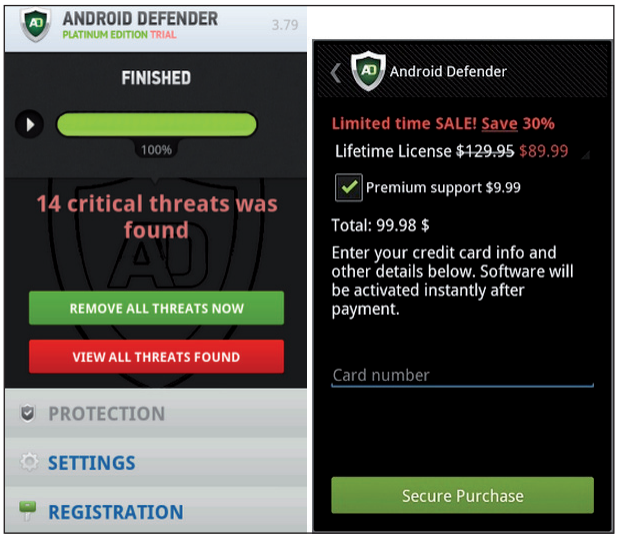

Figure 1: Screenshots of the Android Defender FakeAV.

AN ANALYSIS OF SIMPLELOCKER

To begin *Android* reverse engineering you will need the following tools: *Android* SDK [5], *dex2jar* [6], *apktool* [7], *Java Decompiler* [8], and an archiver on a *Windows* machine.

First, I suggest starting an emulator and launching SimpleLocker (MD5: fd694cf5ca1dd4967ad6e8c67241114c) to see how it works. To verify its supposed functionality we need to copy any picture or document to the SD card to be encrypted by the cryptolocker via *Android Debug Bridge* (adb) [9]:

```
adb push pic.png /sdcard/Pictures/
```

Then we install the malware:

```
adb install Android_ransom.apk
```

Once installed, we see a new app, called 'Sex xonix'.

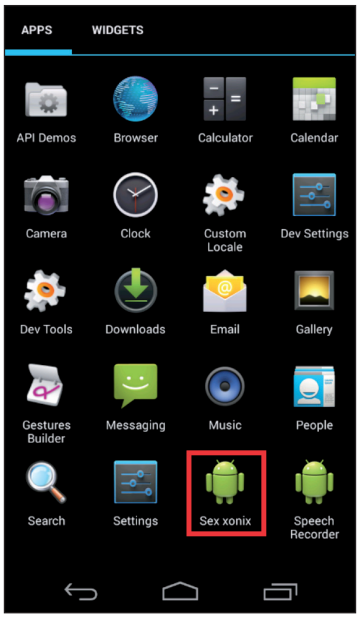

Figure 2: SimpleLocker installs the 'Sex xonix' app.

Let us launch the application and see what happens. We can see the locked screen with the instruction in Russian, as shown in Figure 3.

Now, we can check our files on the SD card to see if they have been encrypted:

```
root@generic:/sdcard/Pictures # ls -al
ls -al
----rwxr-x system   sdcard_rw    28768 2015-06-05
07:39 pic.png.enc
```

The content of the file was encrypted – Figure 4 shows the original and the encrypted file contents.

We have proved that the locker works as it was supposed to, so we can start reverse engineering and looking into the code. The first steps are to unpack the APK and convert the dex file to jar:

```
dex2jar.bat classes.dex
```

Then we can open the jar file in *Java Decompiler* to see the generated Java pseudocode (see Figure 5). After a cursory

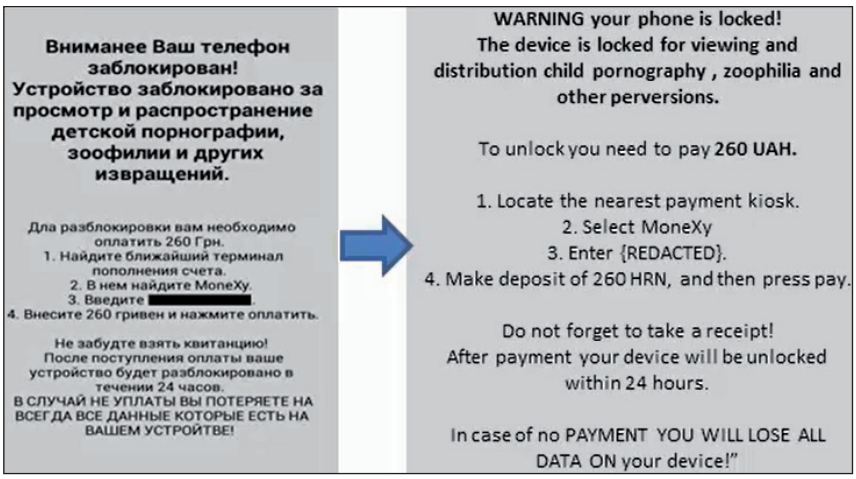

Figure 3: The screen locked by SimpleLocker.

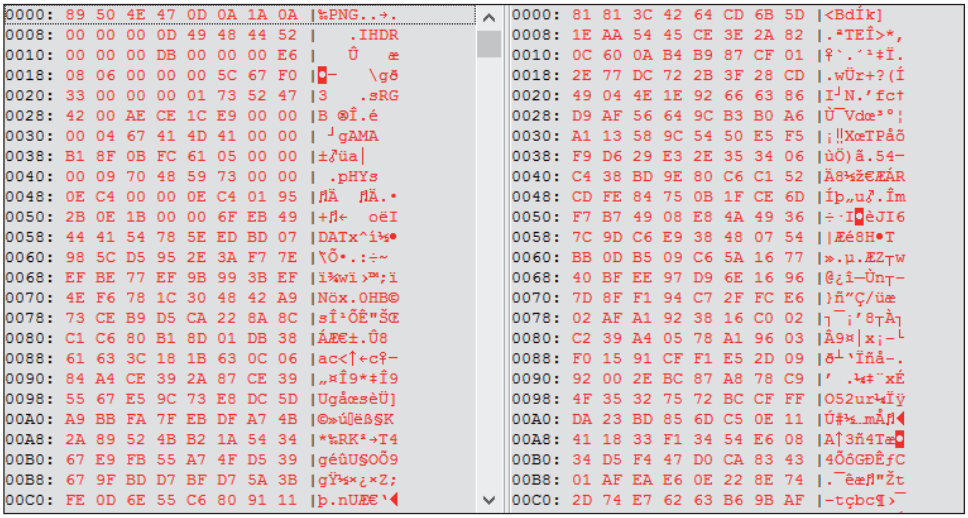

Figure 4: Original (left) and encrypted (right) file contents.

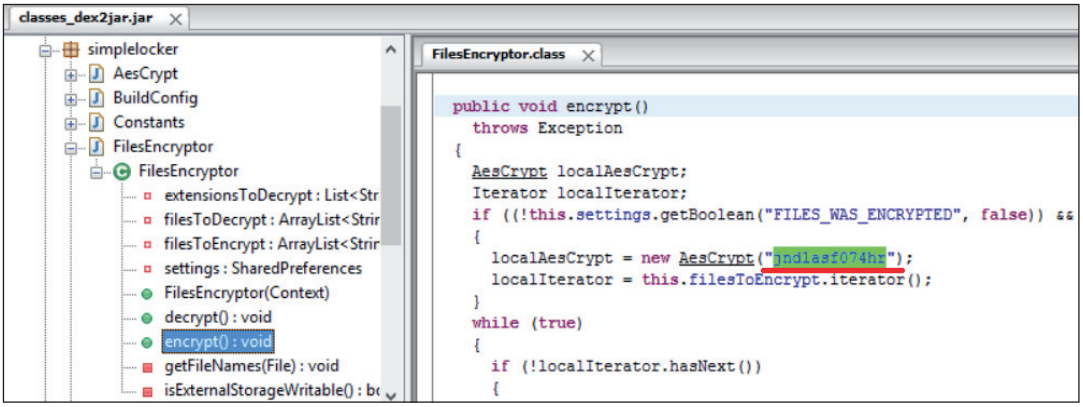

Figure 5: Java pseudocode of SimpleLocker.

overview we can figure out the place where the locker encrypts files and stores the key.

We see a key used to encrypt found files using the AES algorithm.

It is worth mentioning that storing keys inside the code as a string constant is typical for the *Android* botnet malware Wroba.I as well. [10]

You can find even more 'Indicators-of-Compromise' (IoCs) when looking into the class with constants such as C&C server and extensions of files to encrypt:

```
public class Constants
{
  public static final String ADMIN_URL = "http://
xeyocsu7fu2vjhxs.onion/";
  public static final int CHECK_MAIN_WINDOW_TIME_
SECONDS = 1;
  public static final String CIPHER_PASSWORD =
"jndlasf074hr";
  public static final String CLIENT_NUMBER = "19";
  public static final String DEBUG_TAG = "DEBUGGING";
  public static final String DISABLE_LOCKER =
"DISABLE_LOCKER";
  public static final List<String> EXTENSIONS_TO_
ENCRYPT;
  public static final String FILES_WAS_ENCRYPTED =
"FILES_WAS_ENCRYPTED";
  public static final int MONEYPACK_DIGITS_NUMBER =
14;
  public static final int PAYSAFECARD_DIGITS_NUMBER =
16;
  public static final int POLLING_TIME_MINUTES = 3;
  public static final String PREFS_NAME = "AppPrefs";
  public static final int UKASH_DIGITS_NUMBER = 19;

  static
  {
    String[] arrayOfString = new String[13];
    arrayOfString[0] = "jpeg";
    arrayOfString[1] = "jpg";
    arrayOfString[2] = "png";
    arrayOfString[3] = "bmp";
    arrayOfString[4] = "gif";
    arrayOfString[5] = "pdf";
    arrayOfString[6] = "doc";
    arrayOfString[7] = "docx";
    arrayOfString[8] = "txt";
    arrayOfString[9] = "avi";
    arrayOfString[10] = "mkv";
    arrayOfString[11] = v3gp";
    arrayOfString[12] = "mp4";
    EXTENSIONS_TO_ENCRYPT = Arrays.
asList(arrayOfString);
  }
}
```

Under a dalvik debugger we can trace the location where the cryptolocker starts searching through files (Figure 6), and see the files picked for encryption (Figure 7).

MAKING CRYPTO UNLOCKER

This section will be presented at VB2015 in Prague.

CONCLUSION

We see the growing number of ransomware for mobile platforms during the last several years adopting well-known tricks that have been tested on a *Windows* platform. Such programs are simple and have no advanced self-protection mechanisms, so they can be created by anyone with basic programming skills, or even without any. Recently, security researchers from *McAfee* [11] reported on the free Ransomware-as-a-Service called 'Tox Crypto-Malware Kit', which quickly became a popular tool for the creation of *Windows* cryptolockers. It uses basic social engineering techniques to trick users with .scr files as well, as Tor and Bitcoin to provide some level of anonymity. This is the first case that may become a trend not only for *Windows*, but for mobile platforms as well.

REFERENCES

[1] Reveton/FBI ransomware – exposed, explained and eliminated. Sophos, 2012. https://nakedsecurity.sophos.com/2012/08/29/reveton-ransomware-exposed-explained-and-eliminated/.

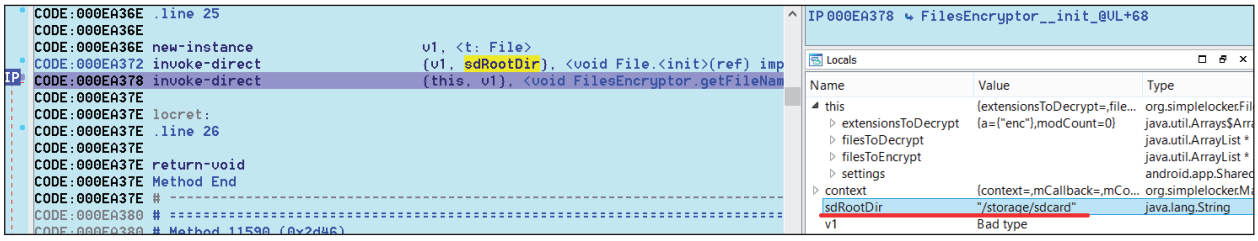

Figure 6: The root directory is initialized with '/storage/sdcard' to search through the files.

Figure 7: The file for encryption has been chosen.

[2] Behind the 'AndroidOS.Koler' distribution network. Kaspersky Labs' Global Research & Analysis Team, 2014. https://securelist.com/blog/research/65189/behind-the-android-os-koler-distribution-network/.

[3] Hamada, J. FakeAV holds Android Phones for Ransom. Symantec, 2013. http://www.symantec.com/connect/blogs/fakeav-holds-android-phones-ransom.

[4] Nigam, R. A Day in the Life of a Mobile Ransomware. Fortinet, 2013. http://blog.fortinet.com/post/a-day-in-the-life-of-a-mobile-ransomware.

[5] Android SDK download. https://developer.android.com/sdk/index.html.

[6] Dex2Jar download. https://github.com/pxb1988/dex2jar.

[7] apktool download. http://ibotpeaches.github.io/Apktool/.

[8] Java Decompiler download. https://code.google.com/p/innlab/downloads/detail?name=jd-gui-0.3.3.windows.zip&.

[9] Android debug bridge. https://developer.android.com/tools/help/adb.html.

[10] Nigam, R. A timeline of mobile botnets. Fortinet, 2015. https://www.virusbtn.com/virusbulletin/archive/2015/03/vb201503-mobile-botnets.

[11] Walter, J. Meet 'Tox': Ransomware for the Rest of Us. McAfee, 2015. https://blogs.mcafee.com/mcafee-labs/meet-tox-ransomware-for-the-rest-of-us.