Malware Analysis Report

Master Degree in Cybersecurity - Dependability Course

Andrea Giuliani, Francesco Rubino, Marc Hofmann

Analysis Tools and Malwares

VirusTotal: platform to analyze suspicious files, domains, IPs and URLs to detect malware and other breaches, automatically share them with the security community.
JD-GUI: a standalone graphical utility that displays Java source codes of ".class" files
MobSF: application permits static and dynamic analysis of APK files.
Android Studio: provides application builders and dynamic analysis of APK files.
Genymotion: Another emulator we used.

MD5 hashes of malwares for identification given by VirusTotal:

- FILE1 - 96c2b215bc929fca8b7651e749d4a6e7
- FILE2 - d65dcf5632685db88e2580ea34801d8c
- FILE3 - 197548d346bd852724de6e690d502e0b
- FILE4 - 0e91ebbcceb761c64d7d7b8bc5889369
- FILE5 - 0289464478c650117ca6d23780583c71

Initial Observations and Working Choices

From VirusTotal, we have immediately observed similarity with FILE 2, FILE 3, FILE 4, FILE 5 and we have decided to start our analysis from FILE 1, which is different from the others. One of the things we notice about FILE 1 is that it is recognized as Dnotua. Dnotua appears to be a type of Adware Malware. So in this case, we should expect that the program displays unwanted ads. Our suspicion is confirmed below, analyzing the code by static means and through the dynamic analysis, observing the results on the user interface of this malware.

Antimalware Analysys

According to VirusTotal the majority of security vendors involved in the analysis have flagged FILE 1 as malicious, and in general categorize it as one of these three categories:

- **PUA**: Potentially Unwanted Application that may be alone or inside legitimate free software program as package.
- **TROJAN**: software that looks legitimate but can take control of the system
- **ADWARE**: Advertising-supported software generates automatically unwanted advertisements in the user interface

There is also a reference on the common vulnerability **CVE-2009-1157** about memory leak on Cisco Adaptive Security Appliances that allows remote attackers to cause a denial of service crafting TCP packets.

The manifest of the APK shows 2 critical permissions requests by the file:

1) **android.permission.ACCESS_NETWORK_STATE**

   Allow the app to view informations about network connections such as which networks exist and are connected

2) **android.permission.INTERNET**

   Allow the app to create network sockets and use custom network protocols

The globally accessible knowledge base of adversary tactics and techniques based on real-world observations (**Mitre ATT&CK**) gives these actions, performed by the malware, as well known behaviors:

- Command and Control
  - Application Layer Protocol
  - Non-Application Layer Protocol
  - Non-Standard Port
  - Encrypted Channels
- Defense Evasion
  - Delete Device Data
- Discovery
  - System Network Connections Discovery
- Impact
  - Delete Device Data
- Collection
  - Network Information Discovery

In dynamic analysis sandboxes, performed automatically by VirusTotal the network traffic generated by the APK matches some rules of IDS, using SNORT, PROOFPOINT and SURICATA:

- 2 LOW rules matched on TCP as **protocol-command-decode**
- 1 INFO rule matched as **potential corporate privacy violation**
- 1 LOW rule matched on a 3 way handshake with wrong ACK package as **generic-protocol-command-decode**

JD-GUI Static Analysis

- **android.support**

Just provides the android support library. It is a set of libraries that backports newer features to older versions of Android or provides utility functions that aren't part of the standard Android framework. This allows developers to use these features while maintaining backward compatibility with older versions of Android.

- **ir.game.co**

This appears to be the more interesting part of the application. This is custom created and has a variety of interesting classes that should be reviewed and analyzed.

- **AudioPlayer.class**

This class simply plays an audio player and provides a progress bar of the .mp3 being played.

- **UpdateActivity.class**

This appears to be the main issue of the App. It looks like it contains an updater function that enables the application to download other APKs, which could of course be a security risk. E.g. we should assume that this application serves as a backdoor or trojan horse. In a sense what this function does, is to download a new APK and update itself from a server. This is problematic because it bypasses the Google Play Store and the user has no possibility of verifying if the update is malicious or not.
The URL for the new APK gets taken from a JSONObject. This JSON object includes the Version of the APK that is to be downloaded, the URL and 'NewDetails':
We suspect that the UrlAPK is retrieved via an RSS feed. We can notice that the created instance is wrongly written. jSONObject, which is lower case at the beginning. It appears, this jSONObject gets retrieved in the following class, where it is also spelled wrongly.

- **tablighActivity.class**

It also appears that this application pulls advertising from the website gamejoo.com or bejo.ir. So as VirusTotal confirms, this app also serves as adware. While advertising itself is not problematic, today's advertising on android apps usually happens via Google's own advertisement program (AdSense).
In this case using a third party advertiser makes sense though. gamejoo.com appears to be located in Iran. Since Iran is sanctioned by EU and US governments, western companies are not allowed to conduct business in Iran. Therefore Adsense would not be available in Iran. We also found some Arabic script in this class, which I am assuming is persian.
Nevertheless, using third party advertisers poses certain risks. These advertising could be used for phishing or other scams. Worst case, they could even be used to download new malware disguised as APKs, providing unsuspecting victims click on them.

- **HTMLActivity.class**

One of the things we also consider to be sketchy, is the HTMLActivity.class. This class serves to pull HTML pages from a remote source. Why this happens is unknown to us, but due to the connectivity to a remote server we again should assume that this functionality can be abused to download malicious information that can be used by the app itself, or in a worst case scenario escape the app and be used somewhere else on the phone.

- **RSSAdapter.class**

Our current suspicion is also that since this App downloads an RSS Feed, this feed can be used to update the Link to download new APKs. The RSS feed links to Purl.org.

Dynamic Analysis

Android Studio

We have first run the application using Android Studio. Installing the game reveals it is called “Bazi Sexy”. Which according to google translate means “sexy game” in Persian. This confirms our suspicion that the App was developed for the Iranian market. The logo, as we can see below simply shows 18+ which further mentally prepares me for the content of the game.
To avoid the network traffic to Iran being blocked, we decided to use a VPN that routes via Singapore, in hope of avoiding western internet blockades.

Launching the app we immediately get greeted by spicy content.

It also opens a link to a telegram channel, but that link is not working.
Clicking on the purchase and activation link just tells us that the app was removed from the store.

MobSF

To understand what the app actually tries to do we used MobSF to analyze the malware run on a virtual machine thanks to the Genymotion app. Using the HTTPTool of MobSF we’ve been able to capture the traffic generated by the app. In particular the main server the application tries to communicate with is a certain forush.co, trying also to communicate with a telegram server located in England, probably to send and receive communications through a Telegram channel.
The MobSF Dynamic Analyzer permitted us to also test some TLS/SSL vulnerabilities. In particular, the tests considered are the following.

TLS Misconfiguration: will uncover insecure configurations that allow HTTPS connections bypassing certificate errors or SSL/TLS errors in WebViews. This is equivalent to not having TLS

TLS Pinning/Certificate Transparency Test: will evaluate the applicationìs TLS/SSL hardening controls and will check if the application implements certificate or public key pinning and or certificate transparency.

TLS Pinning/Certificate Transparency Bypass test: tries to bypass certificate or public key pinning and or certificate transparency controls in the application. MobSF can bypass most of the generic implementation.

The results of these tests are represented in the following table.

TESTS	RESULT
TLS Misconfiguration Test	✅
TLS Pinning/Certificate Transparency Test	❌
TLS Pinning/Certificate Transparency Bypass Test	❌
Cleartext Traffic Test	✅

Finally some interaction with internal databases have been detected. The structure of one of the two databases contains tables used to gather information about the users: credit cards, phone numbers, emails, payments data…

OTHER SAMPLES

- **Antimalware Analysis**

VirusTotal flags as malware all other 4 files analyzed in more than half security vendors consulted by the platform and they are classified as trojan adware as the first file. It finds some differences in the contacted ips location and in some functionalities. Ip addresses are located mostly in China and the most interesting functionalities are related to sms handling, execution of OS command and encryption.

- **Static Analysis**

Observing files in JD-GUI we have discovered a more complex structure with a certain obfuscation technique to increase the complexity of a possible static analysis: very high number of folders, with redundant classes and encryption of all strings used by the application.
We have found inside **com.mobile.bumptech.ordinarySDK** folder the SDK class used to encrypt application string with base64 encoding and XOR encryption.

As VirusTotal has said before, we can be sure that the malwares hides its own behaviors with an adult content app due to the packages found with a lot of porn pictures and a lot of files .mp3 named sexy.

MobSF confirmed our suspicion that the four files were equal in different versions, using the comparison tool provided by the software. Only the main application names are different in the four files: permissions, functionalities, images, behaviors are the same.

References:
https://www.fortiguard.com/encyclopedia/virus/7219846