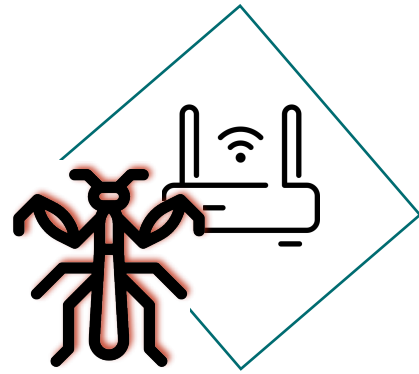




Roaming Mantis

The Art and Science of
detecting Malware with
Quark Engine



Prepared by: JunWei Song, KunYu Chen,
YuShiang Dang, Sin and IokJin Sih

Table of Contents

Introduction	4
Investigating the Android APK.....	5
Summary Report for the APK.....	5
Exploring Malicious Activities	7
Decrypting the DEX File	9
Summary Report of the DEX File.....	10
1. Connect to the remote server	11
2. Start a web server.....	11
3. Monitor/Delete/Send SMS/MMS	12
4. Access phone information.....	13
5. Record audio/video	13
Conclusion	14
Table of Rules Usage.....	15

Table of Figure

Figure 1: Quark-Engine summary report console	5
Figure 2: Rule classification com/svsd/hsMyApplication;onCreate	7
Figure 3: Rule classification com/svsd/hsMyApplication;a	7
Figure 4: Rule classification com/sbddl/gfAdminReceiver\$1;run	8
Figure 5: The file of Roaming Mantis DEX payload	9
Figure 6: Decode Roaming Mantis payload with Base64	9
Figure 7: Quark-Engine summary report for DEX payload	10
Figure 8: Rule classification a/b;a	11
Figure 9: Rule classification b/g;run	11
Figure 10: Rule classification com/n;b	12
Figure 11: Rule classification com/Loader\$s;onReceive	12
Figure 12: Rule classification com/Loader;start	12
Figure 13: Rule classification a/a;a	13
Figure 14: Rule classification com/Loader\$ag\$1;a	13
Figure 15: Rule classification com/j;a	13

Introduction

Roaming Mantis is a notorious malware that was first discovered in 2018 and aims at the Asian region. In the past two years, it has evolved and spread around the world. This malware intends to steal personal sensitive data (e.g. account information, SMS messages and voice calls). Also, the malware bypassed the two-factor authentication by monitoring SMS messages.

According to the report from Kaspersky Lab, the main distribution of this malware is using DNS hijacking through a compromised router. As long as the user connects to the router, their DNS lookup will be redirected to the malicious URL. After the user connects to the malicious website, they will be prompted to download the Google update application, which turns out to be the Android malware.

In this report, we focus on the APK downloaded by victims who were redirected to the malicious URL. We aim at showing how malware analysts can use Quark-Engine to quickly understand what this malware does to the victim.

This malware contains a DEX file encoded by base64 algorithm. Therefore, we will first demonstrate how we can use sets of detection rules to quickly find where to decode the base64 algorithm and where to load the DEX file in the malware.

After decrypting the DEX file. We then further investigate malicious activities in the DEX file with sets of our detection rules. Also, we prove that obfuscation techniques are useless due to the magic design of Quark Engine.

All in all, we show that by using Quark and detection rule sets, malware analysis can be so much fun.

Investigating the Android APK

Summary Report for the APK

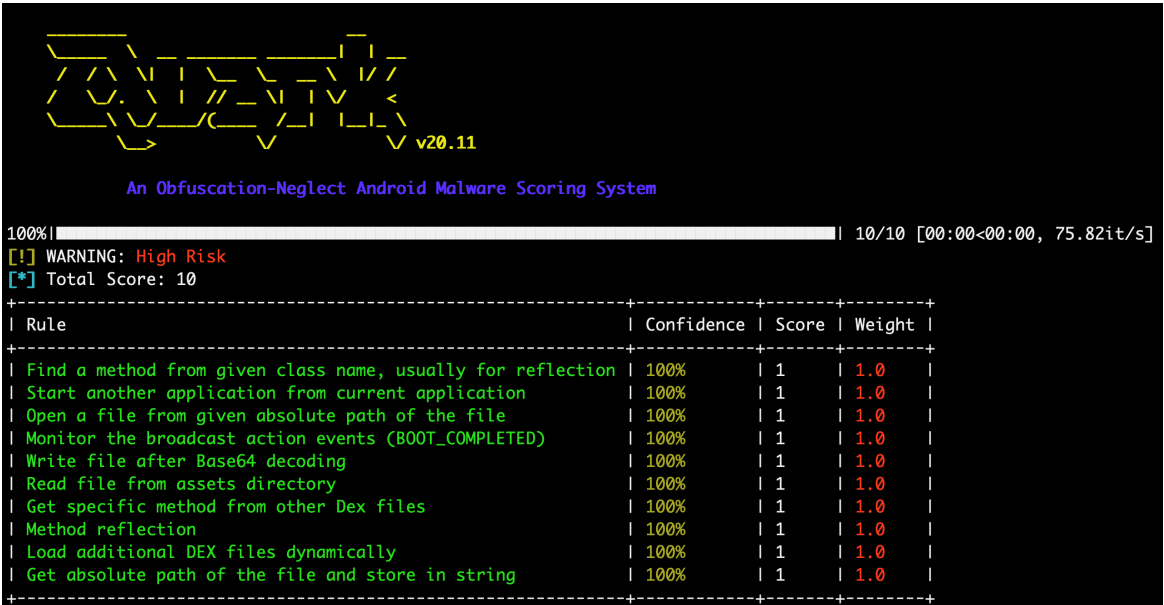
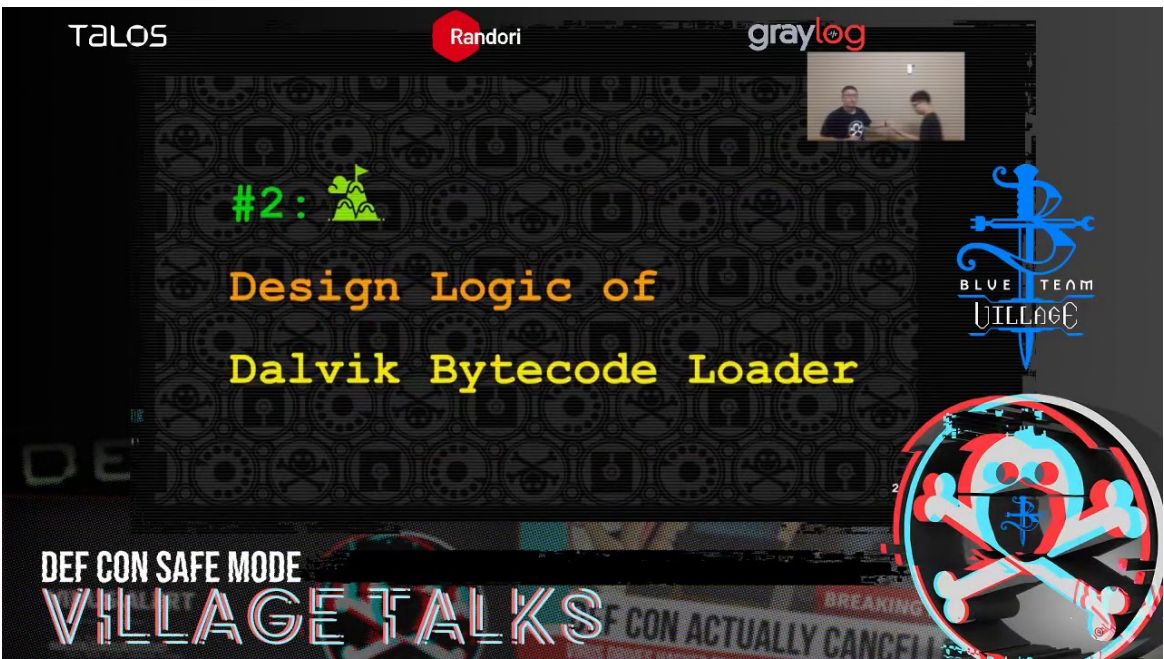


Figure 1: Quark-Engine summary report console

In this report, our engine found 10 potential malicious activities with detection rules accordingly. As for the confidence, scores and weight, please take a look at our talk at DEFCON Blue Team Village videos on YouTube. We explain everything there.

https://www.youtube.com/watch?v=XK-yqHPnsvc&feature=emb_title



The scoring system will only take effect if we have enough detection rules. Before we accumulate enough rules, we set the scores and weights all the same. Therefore, the risk levels and total scores are for reference only.

After generating the summary report, we then use an automatic technique to classify these 10 potential malicious activities.

Exploring Malicious Activities

Rule Classification: onCreate

Parent Function	Lcom/svsd/hsMyApplication;onCreate
Crime Description	<ul style="list-style-type: none">* Open a file from given absolute path of the file* Write file after Base64 decoding* Read file from assets directory* Load additional DEX files dynamically* Get absolute path of the file and store in string

Figure 2: Rule classification com/svsd/hsMyApplication;onCreate

The picture above shows that 5 suspicious activities were found under the function onCreate.

This picture can help malware analysts to understand the malware in an easy way.

As shown above, we found five behaviors in function onCreate. Despite that rules are not listed in the right order, we can still piece them together and tell a story.

With the descriptions in the table, we can simply and quickly guess that this function decodes and load the suspicious payload.

After validating through reading the smali-like source codes, our guess is right!

And the right order of the behaviors is:

1. Get the absolute path of the file and store in a string.
2. Open a file from the given absolute path of the file.
3. Read a file from the assets directory.
4. Write file after Base64 decoding.
5. Load additional DEX files dynamically.

Rule Classification: a

Parent Function	Lcom/svsd/hsMyApplication;a
Crime Description	<ul style="list-style-type: none">* Get specific method from other Dex files* Method reflection

Figure 3: Rule classification com/svsd/hsMyApplication;a

As shown above, it is obvious that function a is about to do method reflection, MAGIC!

Rule Classification: run

Parent Function	Lcom/sbddl/gfAdminReceiver\$1;run
Crime Description	* Find a method from given class name, usually for reflection * Method reflection

Figure 4: Rule classification com/sbddl/gfAdminReceiver\$1;run

Another method reflection detected!!! MAGIC!

Decrypting the DEX File

As mentioned above, we know that the Roaming Mantis reads a file from the assets directory and use Base64 to decode it. For further investigation, we find the file of the DEX payload.

```
(quark-engine) bash-3.2$ unzip Roaming_Mantis.apk -d apk_directory
Archive:  Roaming_Mantis.apk
  inflating: apk_directory/AndroidManifest.xml
  inflating: apk_directory/META-INF/CERT.RSA
  inflating: apk_directory/META-INF/CERT.SF
  inflating: apk_directory/META-INF/MANIFEST.MF
  inflating: apk_directory/META-INF/rxjava.properties
  inflating: apk_directory/assets/db
  inflating: apk_directory/classes.dex
  extracting: apk_directory/res/mipmap-xhdpi-v4/icon.png
  inflating: apk_directory/res/xml/device_admin.xml
  extracting: apk_directory/resources.arsc
(quark-engine) bash-3.2$
```

Figure 5: The file of Roaming Mantis DEX payload

After unzipping the "assets/db" file, we then use the Base64 to decode it and rename it Roaming_Mantis.dex.

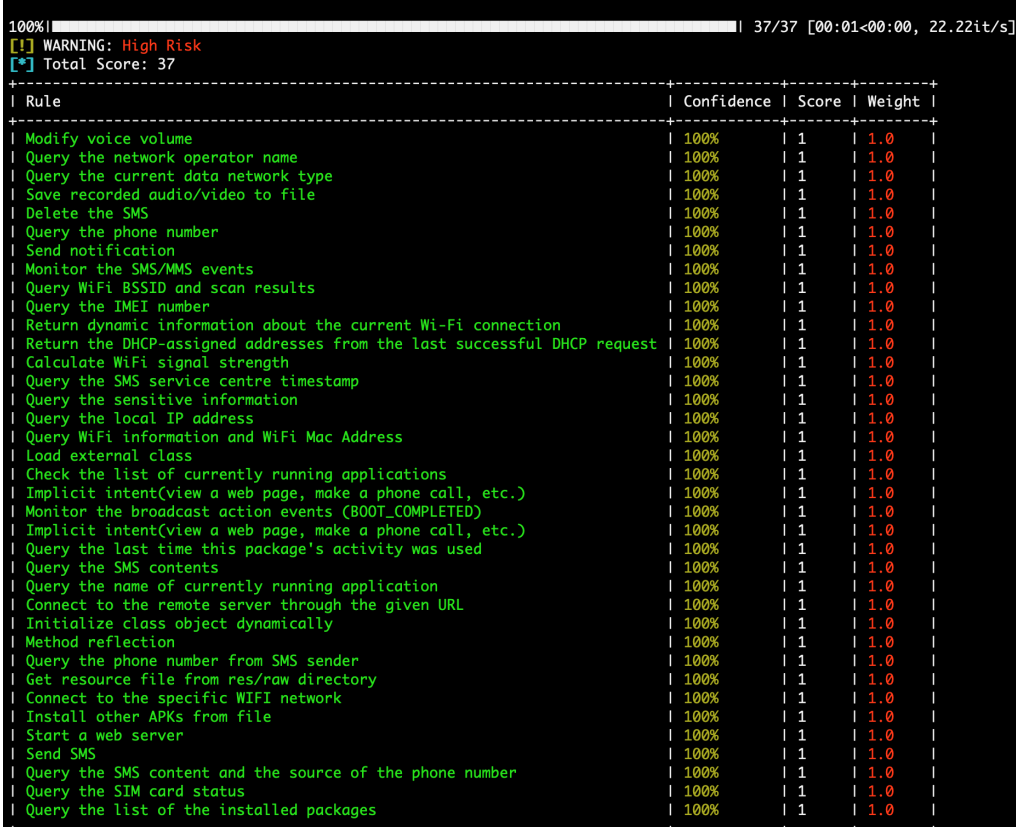
```
(quark-engine) bash-3.2$ cd apk_directory/assets/
(quark-engine) bash-3.2$ base64 --decode db > Roaming_Mantis.dex
(quark-engine) bash-3.2$
```

Figure 6: Decode Roaming Mantis payload with Base64

Now we have the DEX payload!.

Summary Report of the DEX File

Now, let's do the summary report again for the DEX payload, we found 37 suspicious activities.



```
100% | 37/37 [00:01<00:00, 22.22it/s]
[!] WARNING: High Risk
[*] Total Score: 37
```

Rule	Confidence	Score	Weight
Modify voice volume	100%	1	1.0
Query the network operator name	100%	1	1.0
Query the current data network type	100%	1	1.0
Save recorded audio/video to file	100%	1	1.0
Delete the SMS	100%	1	1.0
Query the phone number	100%	1	1.0
Send notification	100%	1	1.0
Monitor the SMS/MMS events	100%	1	1.0
Query WiFi BSSID and scan results	100%	1	1.0
Query the IMEI number	100%	1	1.0
Return dynamic information about the current Wi-Fi connection	100%	1	1.0
Return the DHCP-assigned addresses from the last successful DHCP request	100%	1	1.0
Calculate WiFi signal strength	100%	1	1.0
Query the SMS service centre timestamp	100%	1	1.0
Query the sensitive information	100%	1	1.0
Query the local IP address	100%	1	1.0
Query WiFi information and WiFi Mac Address	100%	1	1.0
Load external class	100%	1	1.0
Check the list of currently running applications	100%	1	1.0
Implicit intent(view a web page, make a phone call, etc.)	100%	1	1.0
Monitor the broadcast action events (BOOT_COMPLETED)	100%	1	1.0
Implicit intent(view a web page, make a phone call, etc.)	100%	1	1.0
Query the last time this package's activity was used	100%	1	1.0
Query the SMS contents	100%	1	1.0
Query the name of currently running application	100%	1	1.0
Connect to the remote server through the given URL	100%	1	1.0
Initialize class object dynamically	100%	1	1.0
Method reflection	100%	1	1.0
Query the phone number from SMS sender	100%	1	1.0
Get resource file from res/raw directory	100%	1	1.0
Connect to the specific WIFI network	100%	1	1.0
Install other APKs from file	100%	1	1.0
Start a web server	100%	1	1.0
Send SMS	100%	1	1.0
Query the SMS content and the source of the phone number	100%	1	1.0
Query the SIM card status	100%	1	1.0
Query the list of the installed packages	100%	1	1.0

Figure 7: Quark-Engine summary report for DEX payload

We simply summarize these suspicious behaviors into twelve categories.

1. Connect to the remote server
2. Start a web server
3. Monitor/Delete/Send SMS/MMS
4. Access network information
5. Access phone information
6. Access personal information
7. Record audio/video
8. Load external class
9. Access currently running applications and installed packages
10. Make a phone call
11. Open a web page
12. Install other APKs from the file

Next, we will introduce some interesting and highly suspicious activities to you based on the above categories.

1. Connect to the remote server

Rule Classification: a/b;a

Parent Function	L a/b;a
Crime Description	* Connect to the remote server through the given URL

Figure 8: Rule classification a/b;a

C2 connections are common in malware. This is a clue for further C2 investigation.

2. Start a web server

Rule Classification: b/g;run

Parent Function	L b/g;run
Crime Description	* Start a web server

Figure 9: Rule classification b/g;run

Our investigation proves that the malware starts a web server and tricks users into filling credentials like username, password, etc.

3. Monitor/Delete/Send SMS/MMS

Rule Classification: com/n;b

Parent Function	Lcom/n;b
Crime Description	* Send SMS

Figure 10: Rule classification com/n;b

Rule Classification: com/Loader\$s;onReceive

Parent Function	Lcom/Loader\$s;onReceive
Crime Description	* Delete the SMS * Query the SMS service centre timestamp * Query the sensitive information * Monitor the broadcast action events (BOOT_COMPLETED) * Query the SMS contents * Query the phone number from SMS sender * Query the SMS content and the source of the phone number

Figure 11: Rule classification com/Loader\$s;onReceive

Rule Classification: com/Loader;start

Parent Function	Lcom/Loader;start
Crime Description	* Monitor the SMS/MMS events * Query the list of the installed packages

Figure 12: Rule classification com/Loader;start

Our investigation proves that these operations concerning "access phone information" might launch activities like:

1. Query the IMEI number so as to targeting the Asian region.
2. Check the SIM card status just make sure the victim's phone works.

4. Access phone information

Rule Classification: a/a;a

Parent Function	L a/a;a
Crime Description	* Query the IMEI number

Figure 13: Rule classification a/a;a

Rule Classification: com/Loader\$ag\$1;a

Parent Function	L com/Loader\$ag\$1;a
Crime Description	* Query the SIM card status

Figure 14: Rule classification com/Loader\$ag\$1;a

Our investigation proves that these operations concerning "access phone information" might launch activities like:

1. Query the IMEI number so as to targeting the Asian region.
2. Check the SIM card status just make sure the victim's phone works.

5. Record audio/video

Rule Classification: com/j;a

Parent Function	L com/j;a
Crime Description	* Save recorded audio/video to file

Figure 15: Rule classification com/j;a

Thrilling! This malware records your audio/video!

Conclusion

This report shows how malware analysts can use the quark engine to quickly guess behaviors of malware and to quickly validate their guess through call graphs and the classification table.

In this report, we show that Quark Engine bypassed the obfuscation techniques used in Roaming Mantis. Also, this time we provide some useful rule sets for the detection. E.g. detecting payload decryption, DEX loader, method reflection, SMS operating, potential c2 connection etc. And all these rules are generated by using our auto-generate tools.

We're proud of our work and we love to play around with it.

So, if you want to take a sip of the quark engine. Please visit our GitHub repository

<https://bit.ly/2ISYG2s>

And the rules used in this report.

<https://bit.ly/3jAOkAv>

You can generate rules by yourself if you can't wait for our next rule release! :D

<https://bit.ly/2IJNxkE>

Table of Rules Usage

You can use the following rules to detect high risk modules listed below.

Rules description	Rule No.
Find a method from given class name, usually for reflection	000019.json
Get absolute path of the file and store in string	000020.json
Load additional DEX files dynamically	000021.json
Open a file from given absolute path of the file	000022.json
Start another application from current application	000023.json
Write file after Base64 decoding	000024.json
Monitor the broadcast action events (BOOT_COMPLETED)	000025.json
Method reflection	000026.json
Get specific method from other DEX files	000027.json
Read file from assets directory	000028.json
Initialize class object dynamically	000029.json
Connect to the remote server through the given URL	000030.json
Check the list of currently running applications	000031.json
Load external class	000032.json
Query the IMEI number	000033.json
Query the current data network type	000034.json
Query the list of the installed packages	000035.json
Get resource file from res/raw directory	000036.json
Send notification	000037.json
Query the phone number	000038.json
Start a web server	000039.json
Send SMS	000040.json
Save recorded audio/video to file	000041.json
Query WiFi BSSID and scan results	000042.json
Calculate WiFi signal strength	000043.json
Query the last time this package's activity was used	000044.json

Rules description	Rule No.
Query the name of currently running application	000045.json
Method reflection	000046.json
Query the local IP address	000047.json
Query the SMS contents	000048.json
Query the phone number from SMS sender	000049.json
Query the SMS service centre timestamp	000050.json
Implicit intent (view a web page, make a phone call, etc.)	000051.json
Delete the SMS	000052.json
Monitor the SMS/MMS events	000053.json
Install other APKs from file	000054.json
Query the SMS content and the source of the phone number	000055.json
Modify voice volume	000056.json
Return the DHCP-assigned addresses from the last successful DHCP request	000057.json
Connect to the specific WIFI network	000058.json
Query the SIM card status	000059.json
Query the network operator name	000060.json
Return dynamic information about the current Wi-Fi connection	000061.json
Query WiFi information and WiFi Mac Address	000062.json
Implicit intent (view a web page, make a phone call, etc.)	000063.json