Roaming Mantis, part V

By Suguru Ishimaru

Published: 2020-02-27 · Archived: 2026-04-05 20:15:51 UTC

Kaspersky has continued to track the Roaming Mantis campaign. The group's attack methods have improved and new targets continuously added in order to steal more funds. The attackers' focus has also shifted to techniques that avoid tracking and research: allowlist for distribution, analysis environment detection and so on. We've also observed new malware families: Fakecop (also known as SpyAgent by McAfee) and Wroba.j (also known as Funkybot by Fortinet).

Distribution of Wroba.g via SMiShing with impersonated brands

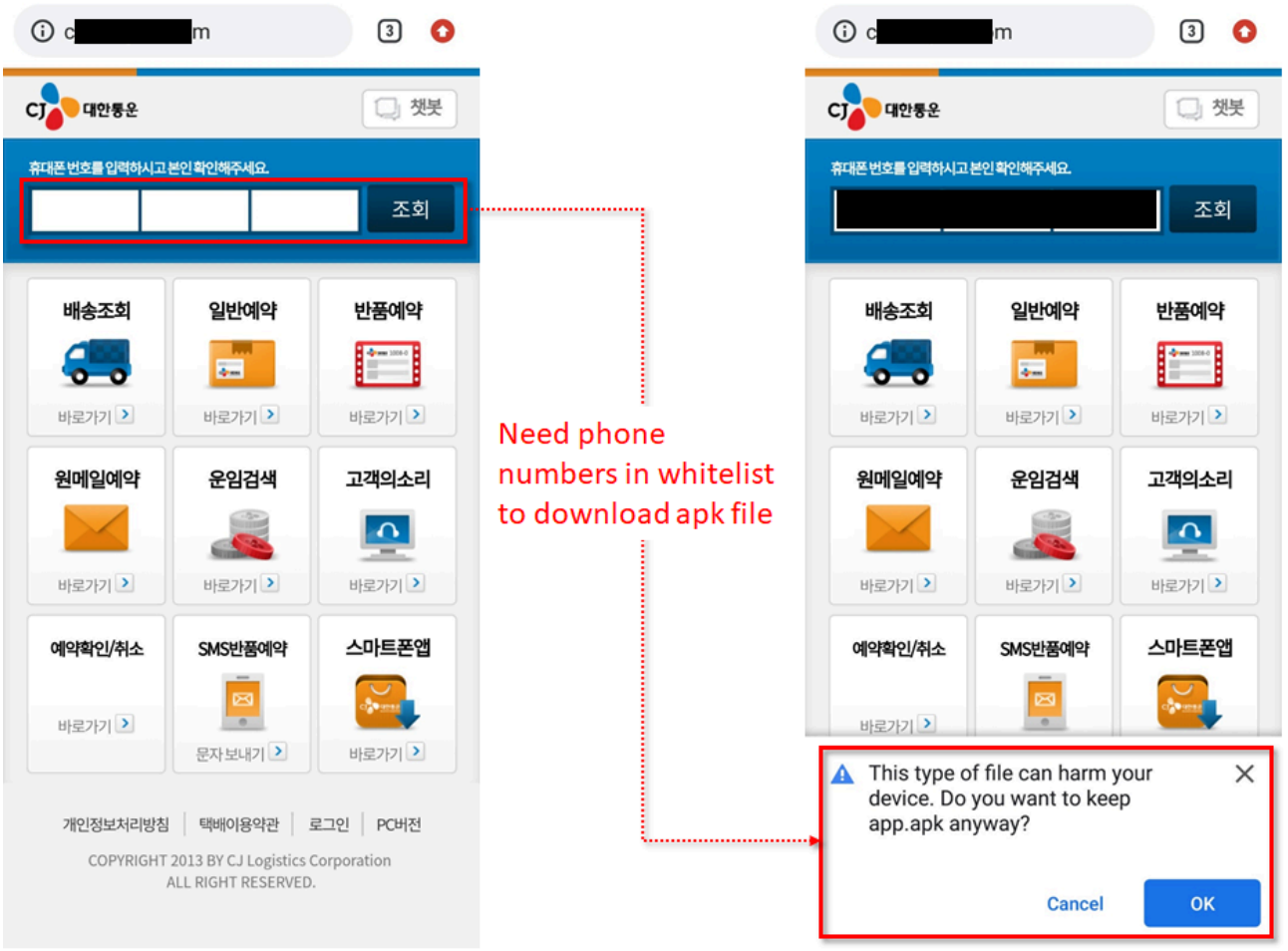
In 2018, the group added a distribution method for Wroba.g (aliases: Moqhao and XLoader), in addition to the original method of DNS hijacking. It was SMiShing using a spoofed delivery notice from a logistics company. In 2019, we confirmed another new method where a downloaded malicious APK file has an icon that impersonates a major courier company brand. The spoofed brand icon is customized for the country it targets, for example, Sagawa Express for Japan; Yamato Transport and FedEx for Taiwan; CJ Logistics for South Korea and Econt Express for Russia.

Examples of SMiShing with Android malware icons impersonating brands

In February 2020, the attacker modified a SMiShing message from a spoofed absence notification to "delivering free masks for the coronavirus issue" in Japan, according to a warning by Japan Cybercrime Control Center (JC3). This once again shows that criminals always make use of hot topics in their activities.

Allowlist feature of Wroba.g landing page for Korea only

The Roaming Mantis actor also employed a new feature in their Wroba.g landing page – currently only on the Korean page. It's a allowlist feature to evade security researchers. When a user visits the landing page, they have to enter their phone number for confirmation. If the phone number is on the allowlist, the landing page distributes a malicious app.apk:

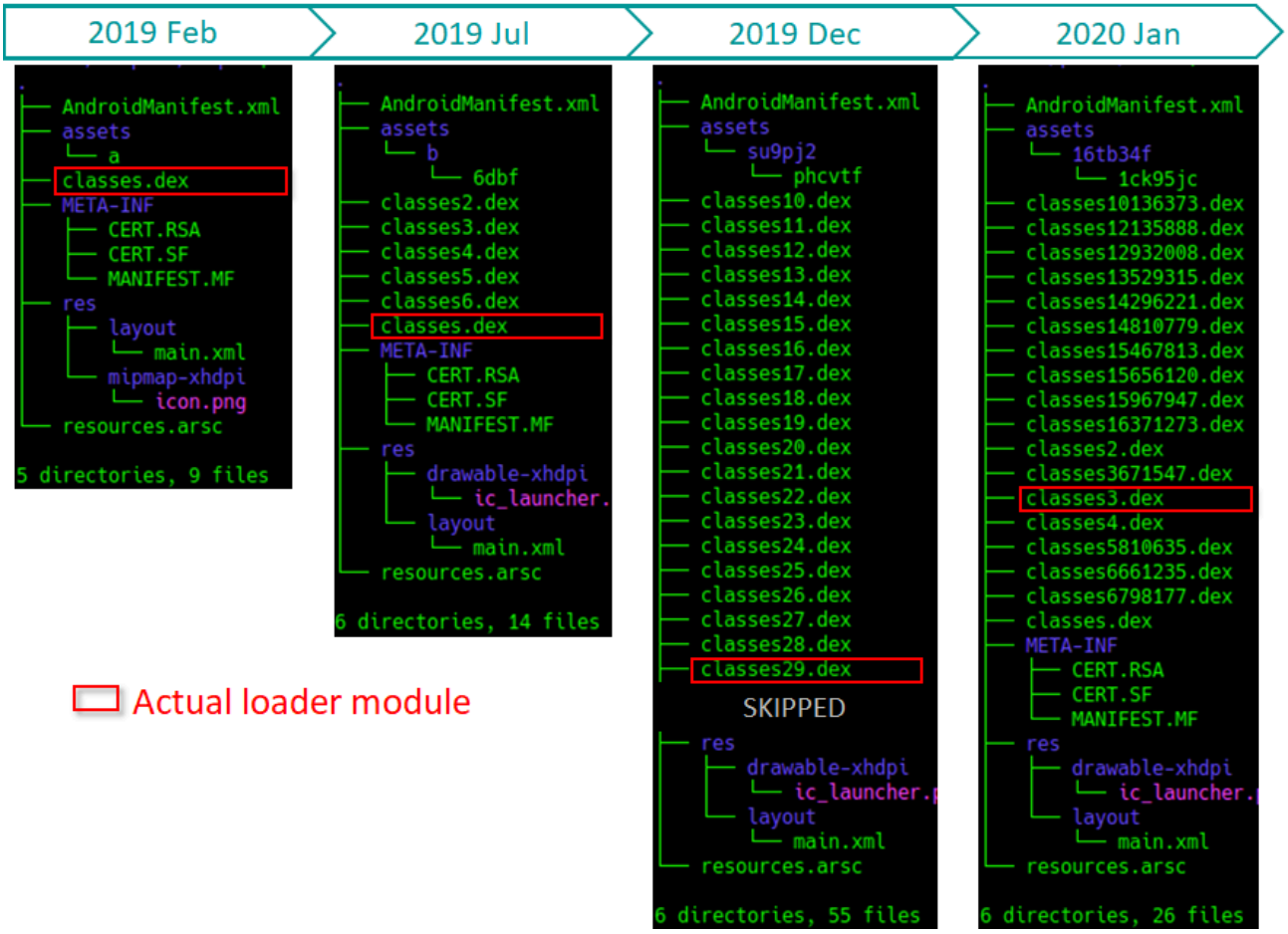


The fake CJ Logistics landing page includes an allowlist

The actor has a habit of trying out their new methods in Korean first. It means the method described above may be applied later on landing pages in other languages as well. If that happens, it would make it almost impossible for researchers to obtain a sample, because it would require a specific phone number in the actor's allowlist database.

Multidex obfuscation trick in a loader module of Wroba.g

A single Dalvik Executable (DEX) has a 64K reference limit. As a workaround, a configuration of Mutidex allows the application to build and read multiple DEX files. In 2019, the actor used Multidex in an APK file to hide a malicious loader module as an obfuscation trick. Our analysis shows that it has been modified little by little:



Transition of obfuscation using Multidex

The classes${num}.dex marked with a red square is the actual malicious loader module. All the other DEX files are simply junk code. However, the encrypted payload of Wroba.g is still under the assets directory and can be decrypted by the simple python script described in our previous blogpost.

Wroba.g is targeting carrier billing and online banks in Japan

The actor has a strong financial motivation. They are targeting carrier billing and online bank accounts. They have implemented redirection to phishing sites to steal credentials in the decrypted payload of Wroba.g:

```
const-string                v0, aPkg # "pkg"
invoke-static               {target_pkg, v0}, <void h.b(ref, ref) h_b@VLL_0> # "jp.co.s[redacted]t"
const-string                v0, aUrl # "url"
invoke-static               {pinterest_url, v0}, <void h.b(ref, ref) h_b@VLL_0> # "https://www.pinterest.com/eme
const-string                v0, aHint # "hint"
invoke-static               {alert_msg, v0}, <void h.b(ref, ref) h_b@VLL_0> # "【[redacted]】お客様がご利用の[redacted]銀行に対
invoke-direct               {this}, <void Object.<init>() imp. @ _def_Object__init_@V>
iput-object                 target_pkg, this, Loader$c_a
iput-object                 pinterest_url, this, Loader$c_b
iput-object                 alert_msg, this, Loader$c_c
```

Hardcoded pkg name, URL of pinterest.com and pop-up message

When the malware detects a specific package of a Japanese online bank or specific mobile carriers on the infected device, it connects in the background to a hardcoded malicious account of pinterest.com to fetch a phishing site with an alert message. The message claims that it has blocked unauthorized access from a third party and asks the

user to click on a button to confirm they want to proceed. If the user clicks the button, they will be redirected to a phishing site:

Redirecting to a phishing site via malicious account on pinterest.com

The targeted packages for online banks and mobile carriers correspond to the relevant accounts on pinterest.com that lead to phishing sites:

Pkgs or mobile carrier	**Accounts on pinterest.com**	**Phishing site in Dec 2019**	**Phishing site in Jan 2020**
jp.co.japannetbank.smtapp.balance	nor**********	jnb.jp-bankq[.]com	N/A
jp.co.jibunbank.jibunmain	abi********	jibun.jp-bankq[.]com	N/A
jp.co.netbk.smartkey.SSNBSmartkey	sin**********	sbi.jp-bankq[.]com	N/A
jp.co.rakuten_bank.rakutenbank	kel**********	rakuten.jp-bankq[.]com	N/A
jp.co.sevenbank.AppPassbook	gh6******	seven.jp-bankq[.]com	N/A

jp.co.smbc.direct	eme**********	smbc.jp-bankq[.]com	smbc.bk-securityo[.]com
jp.japanpost.jp_bank.FIDOapp	fel**********	jppost.jp-bankq[.]com	N/A
jp.mufg.bk.applisp.app	sho**********	mufg.jp-bankq[.]com	N/A
Docomo	ami**********	nttdocomo-uh[.]com	nttdocomo-xm[.]com
au	pos**********	au-ul[.]com	au-xm[.]com
Softbank	ash**********	epos-ua[.]com	N/A

As can be seen in the table above, all the accounts have corresponding phishing sites as of December 2019 (data provided by @ninoseki on Twitter). These destination URLs are continuously changed by the attackers. In January 2020, only three of these accounts were enabled for some reason. However, as it's easy for the criminals to modify the phishing page address, apps without corresponding phishing sites are also likely to be attacked again in the near future.

Wroba.j and Fakecop discovered in 2019

Roaming Mantis has been using Wroba.g and Wroba.f as its main Android malware. In April 2019, we observed two more malware families, Wroba.j and Fakecop. These two malware families have some similarities with the other families in terms of infrastructure, distribution channel, etc. We have created some slides, Roaming Mantis: A melting pot of Android bots in Botconf2019, showing the timeline, impersonated brands, malware features and money laundering method.

Based on our telemetry data, detection rates of both malicious programs were very low. We believe that this was a test by the attacker. However, the most alarming thing we discovered was the following SMS spamming function in Wroba.j:

```
                        # CODE XREF: GrpcMain_sendSms@ILL+FE↑j
if-ne                   v7, v10, loc_27082E
const-string            v6, aSendSmsSuccess # "=======send sms success"
.line 232
invoke-static           {v6}, <void WfkLog.Log(ref) WfkLog_Log@VL>
.line 233
invoke-interface        {v3, v2}, <boolean List.add(ref) imp. @ _def_List_add@ZL>
goto/16                 loc_270986
-------------------------------------------------------------------------------

                        # CODE XREF: GrpcMain_sendSms@ILL:loc_270816↑j
if-ne                   v7, v9, loc_27084A
const-string            v6, aInvalidNumberD # "=======invalid number, delete"
.line 235
invoke-static           {v6}, <void WfkLog.Log(ref) WfkLog_Log@VL>
.line 236
invoke-interface        {v5, v2}, <boolean List.add(ref) imp. @ _def_List_add@ZL>
```

```
new-instance            v0, <t: StringBuilder>
invoke-direct           {v0}, <void StringBuilder.<init>() imp. @ _def_StringBuilder__init_@V>
const-string            v1, aFeedbackSucces # "======feedback, success:"
invoke-virtual          {v0, v1}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
invoke-virtual          {p0}, <ref Object.toString() imp. @ _def_Object_toString@L>
move-result-object      v1
invoke-virtual          {v0, v1}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
const-string            v1, aFailed_0 # ",failed:"
invoke-virtual          {v0, v1}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
invoke-virtual          {p1}, <ref Object.toString() imp. @ _def_Object_toString@L>
move-result-object      v1
invoke-virtual          {v0, v1}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
const-string            v1, aDelete # ",delete:"
invoke-virtual          {v0, v1}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
invoke-virtual          {p2}, <ref Object.toString() imp. @ _def_Object_toString@L>
```

Generating feedback for SMS spamming results

The function automatically creates a sophisticated list of phone numbers from the feedback for SMS spamming results. This malware also has another function that checks the International Mobile Subscriber Identifier (IMSI) to identify mobile carriers in Japan and add the phone number to a relevant spamming list.

```
                        # CODE XREF: PhoneStateUtils_isDocomo@Z+46↑j
.line 250
new-instance            v2, <t: StringBuilder>
invoke-direct           {v2}, <void StringBuilder.<init>() imp. @ _def_StringBuilder__init_@V>
const-string            v3, aImsiNum # "====imsi num:"
invoke-virtual          {v2, v3}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
invoke-virtual          {v2, v0}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
invoke-virtual          {v2}, <ref StringBuilder.toString() imp. @ _def_StringBuilder_toString@L>
move-result-object      v2
invoke-static           {v2}, <void WfkLog.Log(ref) WfkLog_Log@VL>
const-string            v3, a44001 # "44001"
const-string            v4, a44002 # "44002"
const-string            v5, a44003 # "44003"
const-string            v6, a44009 # "44009"
const-string            v7, a4401 # "4401"
```

Check IMSI of mobile carrier

Checking the IMSI of mobile carrier Docomo

According to the hardcoded IMSIs and strings shown below, the attacker seems to be targeting Docomo and Softbank mobile carriers.

IMSI of Docomo:

44001	4401	44058
44002	4402	4406
44003	4403	44087
44009	44049	44099

IMSI of Softbank:

Conclusion

The Roaming Mantis actor is strongly motivated by financial gain and is eager to evade tracking by researchers. It is now employing yet another method – allowlisting – to achieve this. This new method is currently only being applied for Korean pages, but it's only a matter of time before it's implemented for other languages.

The actor is still very active in using SMiShing for Android malware distribution. This is particularly alarming, because it means all infected mobile devices could form a botnet for malware delivery, SMiShing, and so on. ISPs, together with security companies, need to keep a close eye on the Roaming Mantis campaign to understand how to combat it.

Further reading

Further information about the Fakecop and Wroba.j families has also appeared in the following blogs published by McAfee and Fortinet respectively:

- MoqHao Related Android Spyware Targeting Japan and Korea Found on Google Play
- FunkyBot: A New Android Malware Family Targeting Japan

These blogposts provide some interesting updates on Roaming Mantis activities during 2019.

Example of md5 hashes for each APK

e6ae4277418323810505c28d2b6b3647 Wroba.g
939770e5a14129740dc57c440afbf558 Wroba.f
521312a8b5a76519f9237ec500afd534 Wroba.j
6d29caaa8b30cc8b454e74a75d33c902 Fakecop

Source: https://securelist.com/roaming-mantis-part-v/96250/