Android-SpyAgent Threat Report

Date: 26/02/2021

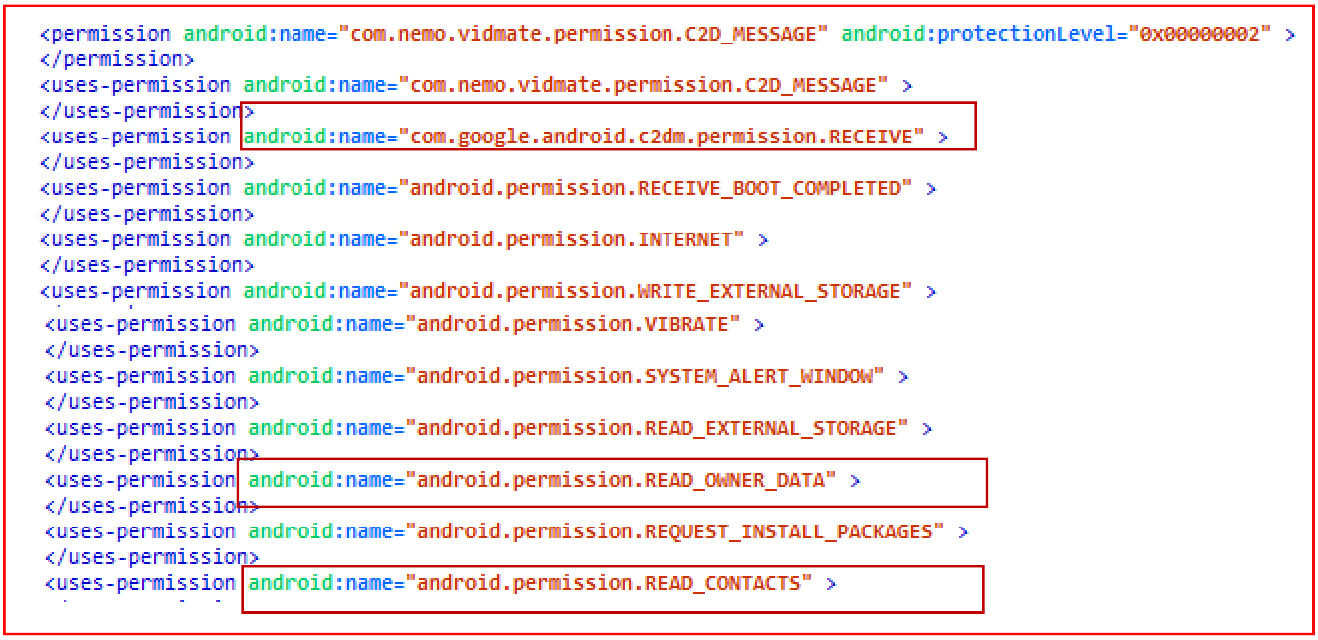
Sanjuktasree Chatterjee

Android applications are nowproviding various functions in one single application. These possess some security risks to the users. Android is a very popular operating system for most users. Cybercriminals are actively targeting this platform and the applicationsto conduct the attack.
In mid of 2020 various popular Chinese applications have been removed from the Play Store. On 24th February 2021, we found a zero-day Android package file in our honeypot which is malicious. This file is a variant of video downloader 'vidmate' which is found to be a spyware application that spies on user's personal data.

File Hash: 1d42c1f1d4eb7e6134c8d14dc752ea40

Technical Analysis:

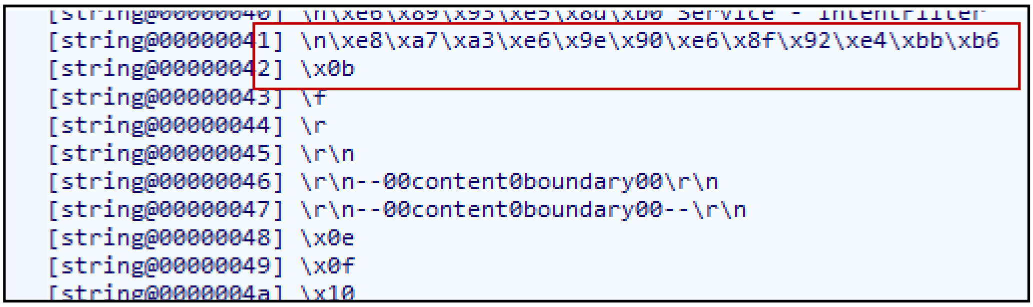
To analyse the android file, first, we need to decompile it. There is a file called Androidmanifest.xml which defines various permissions in the apk file. The file is having excessive user permission that seems suspicious. In Fig: 1, we can see C2D Message permission which is required for receiving messages from the cloud services, the permission for reading owner data and reading contact is not required for this application. These are some malicious characteristics of the spyware which can gather data from the owner.



```
<permission android:name="com.nemo.vidmate.permission.C2D_MESSAGE" android:protectionLevel="0x00000002" >
</permission>
<uses-permission android:name="com.nemo.vidmate.permission.C2D_MESSAGE" >
</uses-permission>
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" >
</uses-permission>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" >
</uses-permission>
<uses-permission android:name="android.permission.INTERNET" >
</uses-permission>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" >
<uses-permission android:name="android.permission.VIBRATE" >
</uses-permission>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" >
</uses-permission>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" >
</uses-permission>
<uses-permission android:name="android.permission.READ_OWNER_DATA" >
</uses-permission>
<uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES" >
</uses-permission>
<uses-permission android:name="android.permission.READ_CONTACTS" >
```

Fig: 1

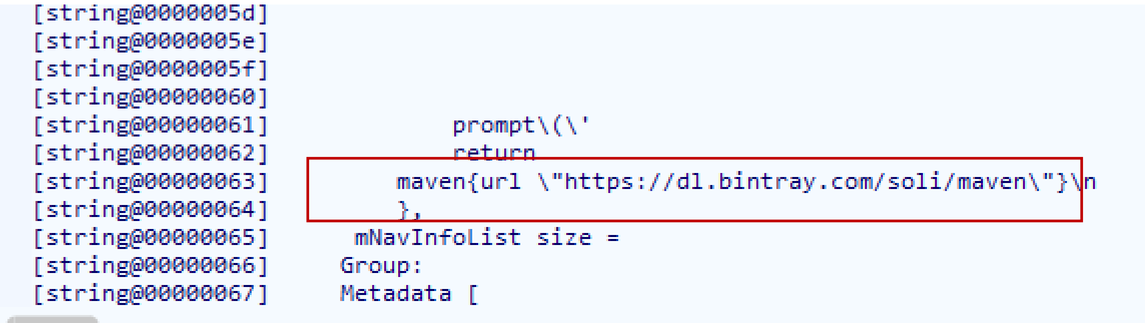
The file has multiple strings, refer to Fig: 2. When we decode the string, it shows a Chinese character that was found in another malicious Chinese android application.



```
[string@00000040] \n\xe6\x89\x93\xe5\x8d\xb0 Service - IntentFilter
[string@00000041] \n\xe8\xa7\xa3\xe6\x9e\x90\xe6\x8f\x92\xe4\xbb\xb6
[string@00000042] \x0b
[string@00000043] \f
[string@00000044] \r
[string@00000045] \r\n
[string@00000046] \r\n--00content0boundary00\r\n
[string@00000047] \r\n--00content0boundary00--\r\n
[string@00000048] \x0e
[string@00000049] \x0f
[string@0000004a] \x10
```

Fig: 2

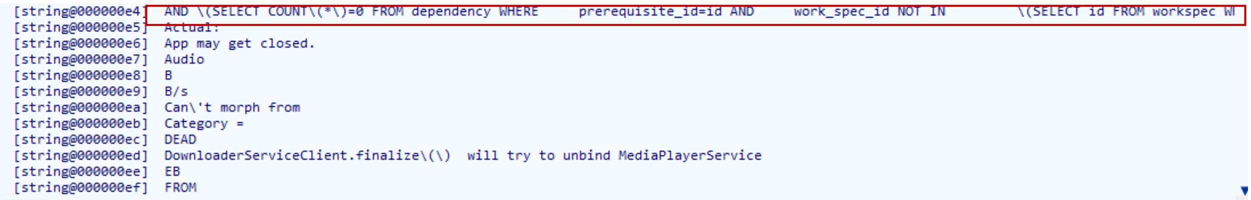
The File contains multiple URLs. Below is one that redirects to a repository that has malicious data.



```
[string@0000005d]
[string@0000005e]
[string@0000005f]
[string@00000060]
[string@00000061]              prompt\(\'
[string@00000062]              return
[string@00000063]          maven{url \"https://dl.bintray.com/soli/maven\"}\n
[string@00000064]          },
[string@00000065]       mNavInfoList size =
[string@00000066]      Group:
[string@00000067]      Metadata [
```

Fig: 3

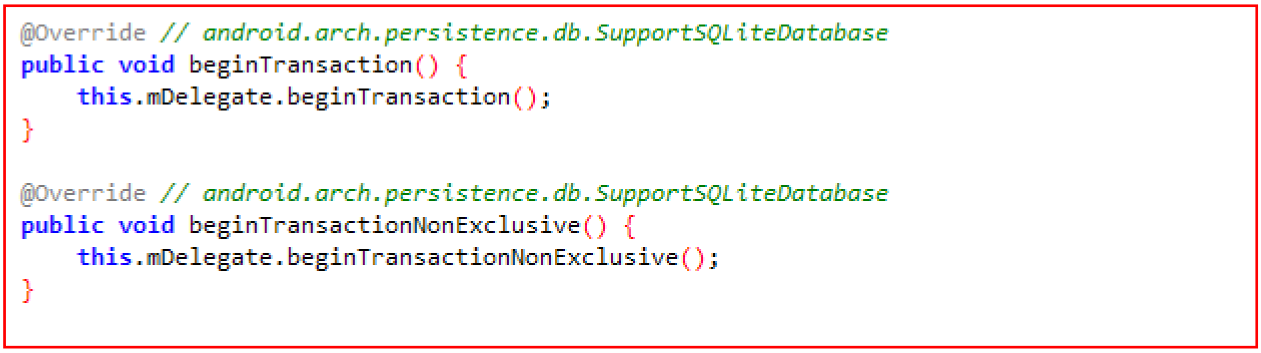
Here is MySQL query that is used to gather data from the workspace. The data gathered by the malware can also be transferred through the `http` response channel which may work in the background.



```
[string@000000e4]  AND \(SELECT COUNT\(*\)=0 FROM dependency WHERE     prerequisite_id=id AND     work_spec_id NOT IN         \(SELECT id FROM workspec WI
[string@000000e5]  Actual:
[string@000000e6]  App may get closed.
[string@000000e7]  Audio
[string@000000e8]  B
[string@000000e9]  B/s
[string@000000ea]  Can\'t morph from
[string@000000eb]  Category =
[string@000000ec]  DEAD
[string@000000ed]  DownloaderServiceClient.finalize\(\)  will try to unbind MediaPlayerService
[string@000000ee]  EB
[string@000000ef]  FROM
```

Fig: 4

The system calls support above MySQL query.

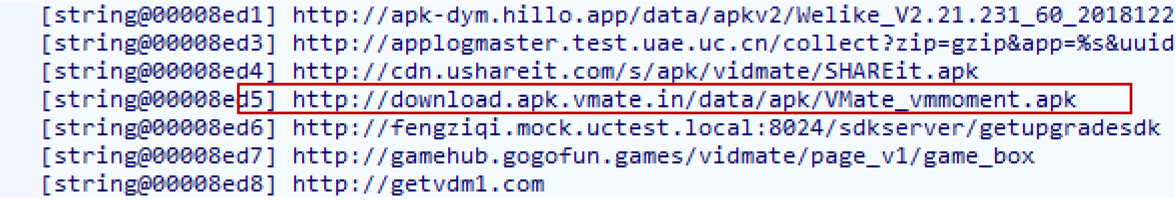


```
@Override // android.arch.persistence.db.SupportSQLiteDatabase
public void beginTransaction() {
    this.mDelegate.beginTransaction();
}

@Override // android.arch.persistence.db.SupportSQLiteDatabase
public void beginTransactionNonExclusive() {
    this.mDelegate.beginTransactionNonExclusive();
}
```

Fig: 5

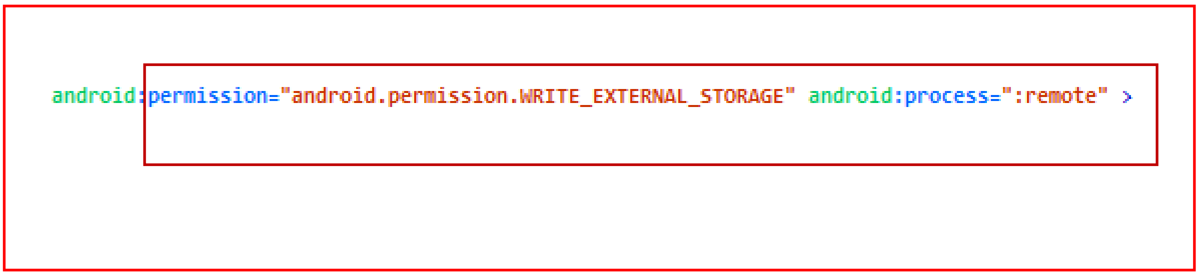
This is one another URL that uses ‘`http`’ request to download an `apk` which is a malicious file.



```
[string@00008ed1] http://apk-dym.hillo.app/data/apkv2/Welike_V2.21.231_60_2018122
[string@00008ed3] http://applogmaster.test.uae.uc.cn/collect?zip=gzip&app=%s&uuid
[string@00008ed4] http://cdn.ushareit.com/s/apk/vidmate/SHAREit.apk
[string@00008ed5] http://download.apk.vmate.in/data/apk/VMate_vmmoment.apk
[string@00008ed6] http://fengziqi.mock.uctest.local:8024/sdkserver/getupgradesdk
[string@00008ed7] http://gamehub.gogofun.games/vidmate/page_v1/game_box
[string@00008ed8] http://getvdm1.com
```

Fig: 6

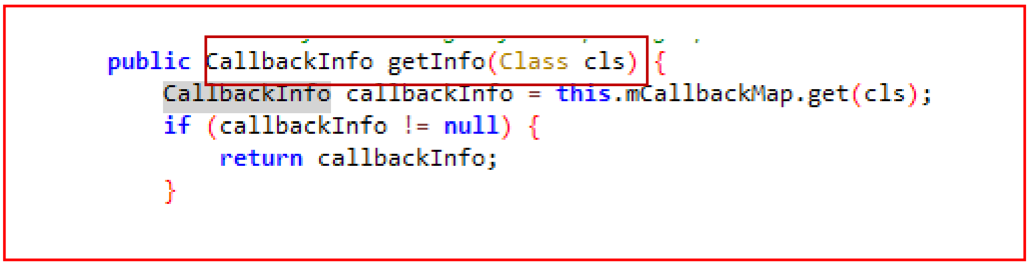
There is permission given to the remote process that can write data to the external storage of the device.



```
android:permission="android.permission.WRITE_EXTERNAL_STORAGE" android:process=":remote" >
```

Fig: 7

Some more information that is embedded in the file's source code, the application will gather the system call-back information from the user's device.



```
public CallbackInfo getInfo(Class cls) {
    CallbackInfo callbackInfo = this.mCallbackMap.get(cls);
    if (callbackInfo != null) {
        return callbackInfo;
    }
```

Fig: 8

IOCS:

Malicious URLs:

http://download.apk.vmate.in/data/apk/VMate_vmmoment.apk
https://d1.bintray.com/soli/maven

MITRE Techniques:

Install Insecure or Malicious Configuration(T1478)
Masquerade as Legitimate Application(T1444)
Access Contact List(T1432)
Access Sensitive Data in Device Logs(T1413)

Subex Secure Protection

Subex Secure detects the malware as "SS_Gen_Andro_Spy_AA" .

Our Honeypot Network

This report has been prepared from the threat intelligence gathered by our honeypot network. This honeypot network is today operational in 62 cities across the world. These cities have at least one of the following attributes:

- Are landing centers for submarine cables
- Are internet traffic hotspots
- House multiple IoT projects with a high number of connected endpoints
- House multiple connected critical infrastructure projects
- Have academic and research centers focusing on IoT
- Have the potential to host multiple IoT projects across domains in the future

Over 3.5 million attacks a day is being registered across this network of individual honeypots. These attacks are studied, analyzed, categorized, and marked according to a threat rank index, a priority assessment framework that we have developed within Subex. The honeypot network includes over 4000 physical and virtual devices covering over 400 device architectures and varied connectivity mediums globally. These devices are grouped based on the sectors they belong to for purposes of understanding sectoral attacks. Thus, a layered flow of threat intelligence is made possible.