APK	ID	Source Signature	Source Category	Sink Signature	Sink Category	Description
backflash	1	android.telephony.TelephonyManager: java.lang.String getSimCountryIso()	LOCATION INFORMATION	java.io.OutputStreamWriter: void write(java.lang.String)	FILE	This malicious flow leaks the SIM country code into an OutputStream, writing to data that is later read by following flow.
backflash	2	android.content.Context: java.io.FileInputStream openFileInput(java.lang.String)	FILE	java.net.HttpURLConnection: java.io.OutputStream getOutputStream()	NETWORK	The second part of the above partial flow that leaks the country code into a URL that is then retrieved. Note: execute(....) internally calls doInBackground(...).
backflash	3	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.DataOutputStream: void write(byte[],int,int)	NETWORK	This malicious flow sends the telephone number of incoming calls via file upload to a remote server.
backflash	4	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.net.HttpURLConnection: java.io.OutputStream getOutputStream()	NETWORK	This malicious flow reads incoming number and appends it in a URL as parameter for a HTTP POST request to a remote server.
backflash	5	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.OutputStreamWriter: void write(java.lang.String)	NETWORK	This malicious flow writes phone number to a file which will be posted to a remote server.
backflash	6	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.OutputStreamWriter: void write(java.lang.String)	NETWORK	This malicious flow writes phone number to a file which will be posted to a remote server.
backflash	7	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.OutputStreamWriter: void write(java.lang.String)	NETWORK	This malicious flow writes phone number to a file which will be posted to a remote server.
backflash	8	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.OutputStreamWriter: void write(java.lang.String)	NETWORK	This malicious flow writes phone number to a file which will be posted to a remote server.
backflash	9	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.OutputStreamWriter: void write(java.lang.String)	NETWORK	Negative flow.
backflash	10	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.OutputStreamWriter: void write(java.lang.String)	NETWORK	Negative flow.
backflash	11	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.net.HttpURLConnection: java.io.OutputStream getOutputStream()	NETWORK	Negative flow.
backflash	12	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.net.HttpURLConnection: java.io.OutputStream getOutputStream()	NETWORK	Negative flow.
backflash	13	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.net.HttpURLConnection: java.io.OutputStream getOutputStream()	NETWORK	Negative flow.
backflash	14	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.net.HttpURLConnection: java.io.OutputStream getOutputStream()	NETWORK	Negative flow.
backflash	15	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.net.HttpURLConnection: java.io.OutputStream getOutputStream()	NETWORK	Negative flow.
backflash	16	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.net.HttpURLConnection: java.io.OutputStream getOutputStream()	NETWORK	Negative flow.
backflash	17	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.OutputStreamWriter: void write(java.lang.String)	NETWORK	This malicious flow writes incoming phone number to a file which will be posted to a remote server.
backflash	18	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.OutputStreamWriter: void write(java.lang.String)	NETWORK	Negative flow.
backflash	19	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.DataOutputStream: void write(byte[],int,int)	NETWORK	This malicious flow writes phone number to a file and the file content is posted to a remote server.
backflash	20	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.DataOutputStream: void write(byte[],int,int)	NETWORK	This malicious flow writes phone number to a file and the file content is posted to a remote server.
backflash	21	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.DataOutputStream: void write(byte[],int,int)	NETWORK	This malicious flow writes phone number to a file and the file content is posted to a remote server.
backflash	22	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.DataOutputStream: void write(byte[],int,int)	NETWORK	This malicious flow writes phone number to a file and the file content is posted to a remote server.
backflash	23	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.DataOutputStream: void write(byte[],int,int)	NETWORK	Negative flow
backflash	24	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	java.io.DataOutputStream: void write(byte[],int,int)	NETWORK	Negative flow
beita_com_beita_contact	1	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	javax.mail.Transport: void sendMessage(javax.mail.Message,javax.mail.Address[])	SMS MMS	This malicious flow leaks contact data via email.
beita_com_beita_contact	2	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.BufferedWriter: void write(java.lang.String)	FILE	This malicious partial flow leaks contact data into a file (that is later uploaded).
beita_com_beita_contact	3	java.io.FileInputStream: void <init	FILE	java.io.DataOutputStream: void write(byte[],int,int)	NETWORK	This malicious partial flow uploads the file containing contact data to a remote server via HTTP POST request.
cajino_baidu	1	java.io.File: java.io.File[] listFiles()	FILE	com.baidu.inf.iis.bcs.BaiduBCS: com.baidu.inf.iis.bcs.response.BaiduBCSResponse putObject(com.baidu.inf.iis.bcs.request.PutObjectRequest)	NETWORK	This malicious flow reads photos and uploads them to a remote server via HTTP request. This malicious behavior will be triggered, when then PushMessageReceiver (a BroadcastReceiver) receives a command which contains the string 'photo'.

cajino_baidu	2	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: void write(java.lang.String)	FILE	This malicious flow reads contacts and saves them to a file. This file will be later uploaded to a remote server (not a part of this flow). This malicious behavior will be triggered, when then PushMessageReceiver (a BroadcastReceiver) receives a command which contains the string 'contact'.
cajino_baidu	3	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: void write(java.lang.String)	FILE	This malicious flow reads call log and saves them to a file. This file will be later uploaded to a remote server (not a part of this flow). This malicious behavior will be triggered, when then PushMessageReceiver (a BroadcastReceiver) receives a command which contains the string 'call_log'.
cajino_baidu	4	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: void write(java.lang.String)	FILE	This malicious flow reads sms messages and saves them to a file. This file will be later uploaded to a remote server (not a part of this flow). This malicious behavior will be triggered, when then PushMessageReceiver (a BroadcastReceiver) receives a command which contains the string 'upload_message'.
cajino_baidu	5	android.location.LocationManager: android.location.Location getLastKnownLocation(java.lang.String)	LOCATION INFORMATION	java.io.FileWriter: void write(java.lang.String)	FILE	This malicious flow reads location information and saves it to a file. This file will be later uploaded to a remote server (not a part of this flow). This malicious behavior will be triggered, when then PushMessageReceiver (a BroadcastReceiver) receives a command which contains the string 'location'.
cajino_baidu	6	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	java.io.FileWriter: void write(java.lang.String)	FILE	This malicious flow reads device Id and saves it to a file. This file will be later uploaded to a remote server (not a part of this flow). This malicious behavior will be triggered, when then PushMessageReceiver (a BroadcastReceiver) receives a command which contains the string 'phone'.
cajino_baidu	7	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	com.baidu.inf.iis.bcs.BaiduBCS: com.baidu.inf.iis.bcs.response.BaiduBCSResponse putObject(com.baidu.inf.iis.bcs.request.PutObjectRequest)	NETWORK	This malicious flow reads device Id and uploads it to a remote server via HTTP request. This malicious behavior will be triggered, when then PushMessageReceiver (a BroadcastReceiver) receives a command which contains the string 'phone'.
cajino_baidu	8	java.io.File: java.io.File[] listFiles()	FILE	java.io.FileWriter: void write(java.lang.String)	FILE	This malicious flow reads file list and saves it to a file. This file will be later uploaded to a remote server (not a part of this flow). This malicious behavior will be triggered, when then PushMessageReceiver (a BroadcastReceiver) receives a command which contains the string 'list_file'.
cajino_baidu	9	java.io.File: void <init	FILE	com.baidu.inf.iis.bcs.BaiduBCS: com.baidu.inf.iis.bcs.response.BaiduBCSResponse putObject(com.baidu.inf.iis.bcs.request.PutObjectRequest)	NETWORK	This malicious flow uploads files. This malicious behavior will be triggered, when then PushMessageReceiver (a BroadcastReceiver) receives a command which contains the string 'upload_file'.
cajino_baidu	10	java.io.File: void <init	FILE	java.io.File: boolean delete()	CRITICAL FUNCTION	This malicious flow delete file. This malicious behavior will be triggered, when then PushMessageReceiver (a BroadcastReceiver) receives a command which contains the string 'delete_file'.
cajino_baidu	11	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	com.baidu.inf.iis.bcs.BaiduBCS: com.baidu.inf.iis.bcs.response.BaiduBCSResponse putObject(com.baidu.inf.iis.bcs.request.PutObjectRequest)	NETWORK	Negative flow. The device id is only used as folder name.
cajino_baidu	12	java.io.File: void <init	FILE	com.baidu.inf.iis.bcs.BaiduBCS: com.baidu.inf.iis.bcs.response.BaiduBCSResponse putObject(com.baidu.inf.iis.bcs.request.PutObjectRequest)	NETWORK	This malicious flow uploads files to a remote server.
cajino_baidu	13	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	com.baidu.inf.iis.bcs.BaiduBCS: com.baidu.inf.iis.bcs.response.BaiduBCSResponse putObject(com.baidu.inf.iis.bcs.request.PutObjectRequest)	NETWORK	Negative flow. The device id is only used as folder name.
cajino_baidu	14	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	com.baidu.inf.iis.bcs.BaiduBCS: com.baidu.inf.iis.bcs.response.BaiduBCSResponse putObject(com.baidu.inf.iis.bcs.request.PutObjectRequest)	NETWORK	Negative flow. The device id is only used as folder name.
cajino_baidu	15	java.io.File: void <init	FILE	java.io.File: boolean delete()	CRITICAL FUNCTION	This malicious flow deletes files to cover up the malicious behaviors.
chat_hook	1	android.content.Context: java.io.FileInputStream openFileInput(java.lang.String)	FILE	java.io.DataOutputStream: void writeBytes(java.lang.String)	CRITICAL FUNCTION	This malicious flow reads mount data to construct a command and execute it.
chat_hook	2	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious partial flow reads chat history and sends it to a remote server per an HTTP POST request.
chat_hook	3	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs device id.
chat_hook	4	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs chat history.
chat_hook	5	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs last message time in chat history.
chat_hook	6	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs chat history.
chat_hook	7	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs imei.
chat_hook	8	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs chat history.

chat_hook	9	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs lat message time and with whom.
chat_hook	10	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs imei.
chat_hook	11	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow is an implicit flow, since reaching sink depends on the data from source.
chat_hook	12	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs package information
chat_hook	13	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	Negative flow.
chulia	1	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow leaks the recipient or sender of an sms text message and posts it to an HTTP Server
chulia	2	android.location.Location: double getLongitude()	LOCATION INFORMATION	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow leaks the location of the android device and posts it to an HTTP Server
chulia	3	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow leaks contacts and posts it to an HTTP server.
chulia	4	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow leaks phone number to a remote server. It reads first read the phone number in AlarmService class, later puts this information into a intent and starts a BroadcastReceiver sendReceiver.
death_ring_materialflow	1	android.telephony.gsm.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	android.telephony.gsm.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This malicious flow reads SMS content to fake text message for fishing.
dsencrypt_samp	1	android.content.res.AssetManager: java.io.InputStream open(java.lang.String)	FILE	java.lang.reflect.Method: java.lang.Object invoke(java.lang.Object,java.lang.Object[])	CRITICAL FUNCTION	This malicious flow reads an encrypted zip file from asset folder, decrypts it and extracts class.dex which contains malicious code. The malicious code is called via reflection.
exprespam	1	android.telephony.TelephonyManager: java.lang.String getLine1Number()	UNIQUE IDENTIFIER	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow leaks the telephone number to a remote server.
exprespam	2	frhfsd.siksdk.ujdsfjkfsd.WrehifsdkjsActivity: android.database.Cursor managedQuery(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow leaks contact data to a remote server.
fakeappstore	1	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This malicious flow reads contacts from the device and spams them with SMS.
fakeappstore	2	org.apache.http.HttpEntity: java.io.InputStream getContent()	INTERNET SOURCE	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This malicous flow reads a message from an HTTP Server and sends a SMS to the retreived numbers (command and control).
fakeappstore	3	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow leaks the device id to an HTTP Server.
fakebank_android_samp	1	com.example.bankmanager.BankActivity: android.view.View findViewById(int)	ACCOUNT INFORMATION	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious ICC flow crosses 4 different activities: BankActivity, BankNumActivity BankScardActivity and BankEndActivity. It starts from reading bank account number from user input in BankActivity; when the user performs a click on a button, the BankNumActivity is started; the BankScardActivity and BankEndActivity are also started by clicking on a button. In the first three activities, bank information such as account number, passwords and TAN number are stored into the static fields of the class BankInfo. There are input validations for user inputs to make sure that only valid bank information is collected. In the last activity, all collected information will be read from BankInfo and written to a List. This list will be sent to a remote server per an HTTP POST request.
fakebank_android_samp	2	android.telephony.TelephonyManager: java.lang.String getSimSerialNumber()	UNIQUE IDENTIFIER	java.net.HttpURLConnection: java.io.InputStream getInputStream()	NETWORK	This malicious flow starts by reading the serial number of the SIM card, then writes to the field of the class BankEndActivity, thento a HashMap. Then information in the HashMap will be appended to a String which will be sent to a remote server per an HTTP GET request.
fakebank_android_samp	3	android.telephony.TelephonyManager: java.lang.String getSimSerialNumber()	UNIQUE IDENTIFIER	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow starts by reading the serial number of the SIM card, then writes to a List. This list will be sent to a remote server per an HTTP POST request.
fakebank_android_samp	4	android.telephony.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow starts by reading SMS message, then writes to a BasicNameValuePair object, then a HashMap. The hashmap will be sent to a rmote server per an HTTP POST request when a Thread created by the class smsReceiver runs.
fakebank_android_samp	5	java.io.File: void <init	FILE	com.example.service.InstallService: void startActivity(android.content.Intent)	INTENT	This malicous flow installs an apk from sdcard.

fakedaum	1	android.telephony.SmsMessage: android.telephony.SmsMessage createFromPdu(byte[])	SMS MMS	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads a received SMS and uploads it to an HTTP Server.
fakedaum	2	android.telephony.TelephonyManager: java.lang.String getSimSerialNumber()	UNIQUE IDENTIFIER	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads the IMEI and uploads it to an HTTP Server.
fakemart	1	android.telephony.gsm.SmsMessage: android.telephony.gsm.SmsMessage createFromPdu(byte[])	SMS MMS	java.net.URLConnection: java.io.InputStream getInputStream()	NETWORK	This malicious flow reads incoming SMSs and posts them to a URL.
fakemart	2	android.telephony.gsm.SmsMessage: android.telephony.gsm.SmsMessage createFromPdu(byte[])	SMS MMS	java.net.URLConnection: java.io.InputStream getInputStream()	NETWORK	This malicious flow reads incoming SMSs and posts them to a URL.
fakeplay	1	android.telephony.TelephonyManager: java.lang.String getNetworkOperatorName()	NETWORK INFORMATION	java.io.OutputStream: void write(byte[],int,int)	EMAIL	This malicious flow leaks telephoney network information via email.
fakeplay	2	android.telephony.TelephonyManager: java.lang.String getLine1Number()	UNIQUE IDENTIFIER	java.io.OutputStream: void write(byte[],int,int)	EMAIL	This malicious flow leaks telephone number via email.
faketaobao	1	android.widget.EditText: android.text.Editable getText()	ACCOUNT INFORMATION	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads user given taobao password and sends to a remote server.
faketaobao	2	android.widget.EditText: android.text.Editable getText()	ACCOUNT INFORMATION	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads user given taobao username and sends to a remote server.
faketaobao	3	android.widget.EditText: android.text.Editable getText()	ACCOUNT INFORMATION	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads user's ID and sends it to a remote server.
faketaobao	4	android.widget.EditText: android.text.Editable getText()	ACCOUNT INFORMATION	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flows reads user given password and sends to a remote server.
godwon_samp	1	android.telephony.TelephonyManager: java.lang.String getLine1Number()	UNIQUE IDENTIFIER	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads the phone number and puts it into a ArrayList. This list is sent to a remote server via a HTTP Post request.
godwon_samp	2	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads the device Id when the phone number is empty. This device Id is stored in a ArrayList. This list is sent to a remote server via a HTTP Post request.
godwon_samp	3	android.content.Intent: java.io.Serializable getSerializableExtra(java.lang.String)	SMS MMS	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow exists in the onReceive callback method of a BroadcastReciver, it reads SMS messages to array, appends all messages to a string, stores it into a List and sends the list to a remote server via a HTTP POST request.
godwon_samp	4	android.content.Intent: java.io.Serializable getSerializableExtra(java.lang.String)	SMS MMS	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow exists in the onReceive callback method of a BroadcastReciver, it reads SMS messages to array, writes the sender's address into a private field, stores it into a List and sends the list to a remote server via a HTTP POST request.
godwon_samp	5	android.telephony.TelephonyManager: java.lang.String getLine1Number()	UNIQUE IDENTIFIER	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads the phone number and sends it to a rmote server via a HTTP Post request.
godwon_samp	6	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads device id when the phone number is empty, and it send the device id to a remote server via a HTTP Post request.
hummingbad_android_samp	1	android.database.Cursor: java.lang.String getString(int)	DATABASE	android.database.sqlite.SQLiteDatabase: long insert(java.lang.String,java.lang.String,android.content.ContentValues)	DATABASE	This malicious ICC flow crosses one activity and two services. It starts by reading advertisement information from data base into a JSONObject, when the user performs a click, new advertisements will be downloaded and saved into data base.
hummingbad_android_samp	2	android.database.Cursor: java.lang.String getString(int)	DATABASE	android.database.sqlite.SQLiteDatabase: int update(java.lang.String,android.content.ContentValues,java.lang.String,java.lang.String[])	DATABASE	This malicious ICC flow crosses one activity and two services. It starts by reading advertisement information from data base into a JSONObject, when the user performs a click, new advertisements will be downloaded and updated in the data base.
jollyserv	1	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	java.util.concurrent.ThreadPoolExecutor: java.util.concurrent.Future submit(java.lang.Runnable)	NETWORK	The malicious flow aves the the phone's device ID in a public static field and leaks the device ID into an HTTP request.
overlaylocker2_android_samp	1	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	This malicious flow reads the device id and uses it as the return value of the method imei() of the class MeSetting. An object of MeSetting is injected into the WebView by addJavascriptInterface. The method imei() can be invoked via JavaScript code.
overlaylocker2_android_samp	2	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	This malicious flow reads the device id and stores it as a field of an TelephonyInfo object. This information will be used as the return value of the method dualsim() of the class MeSystem. An object of MeSystem is injected into the WebView by addJavascriptInterface. The method dualsim() can be invoked via JavaScript code.
overlaylocker2_android_samp	3	java.lang.Class: java.lang.reflect.Method getDeclaredMethod(java.lang.String,java.lang.Class[])	CRITICAL FUNCTION	java.lang.reflect.Method: java.lang.Object invoke(java.lang.Object,java.lang.Object[])	CRITICAL FUNCTION	This malicious flow uses Java Reflection to execute code which gets the service for a selected sim card.

overlaylocker2_android_samp	4	java.lang.Class: java.lang.reflect.Method getDeclaredMethod(java.lang.String,java.lang.Class[])	CRITICAL FUNCTION	java.lang.reflect.Method: java.lang.Object invoke(java.lang.Object,java.lang.Object[])	CRITICAL FUNCTION	This malicious flow uses Java Reflection to execute code which sends SMS messages.
overlaylocker2_android_samp	5	java.io.FileInputStream: void <init	FILE	java.io.DataOutputStream: void flush()	NETWORK	This malicious flow reads file and sends the file to a remote server via an HTTP POST request.
overlaylocker2_android_samp	6	java.io.FileInputStream: void <init	FILE	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	This malicious flow reads file from a given path and and uses it as the return value of the method read() of the class MeFile. An object of MeFile is injected into the WebView by addJavascriptInterface. The method read() can be invoked via JavaScript code.
overlaylocker2_android_samp	7	android.telephony.SmsMessage: android.telephony.SmsMessage createFromPdu(byte[])	SMS MMS	android.content.Context: android.content.ComponentName startService(android.content.Intent)	INTENT	This malicious flow monitors incoming SMSs.
overlaylocker2_android_samp	8	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlaylocker2_android_samp	9	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlaylocker2_android_samp	10	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlaylocker2_android_samp	11	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlaylocker2_android_samp	12	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlaylocker2_android_samp	13	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlaylocker2_android_samp	14	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlaylocker2_android_samp	15	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlaylocker2_android_samp	16	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negavie flow.
overlaylocker2_android_samp	17	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlaylocker2_android_samp	18	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlaylocker2_android_samp	19	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	Negative flow.
overlay_android_samp	1	exts.whats.activities.Cards: android.view.View findViewById(int)	ACCOUNT INFORMATION	exts.whats.activities.Cards: android.content.ComponentName startService(android.content.Intent)	INTENT	This malicious partial ICC flow starts by reading credit card number from user input and writes to a field of the Activity. When the user performs a click on a button, the information stealing starts. The information will be written to a JSONObject which will be stored into an Intent with the name data. With this intent a new SendService which leaks the card number will be started.
overlay_android_samp	2	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	ACCOUNT INFORMATION	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious partial ICC flow reads credit card information from the Intent with which the SendSevice is started and sends it to a remote server per an HTTP POST request.
overlay_android_samp	3	android.content.Intent: android.os.Bundle getExtras()	SMS MMS	android.content.Context: android.content.ComponentName startService(android.content.Intent)	INTENT	This malicious partial ICC flow starts by reading SMS messages from the Intent with which the MessageReceiver is started. The messages will be written to a Map which will be stored into an Intent. With this intent a new SendService which leaks the card number will be started.
overlay_android_samp	4	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	SMS MMS	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious partial ICC flow reads SMS messages from the Intent with which the SendService is started and sents it to a remote server per an HTTP Post request.

overlay_android_samp	5	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	exts.whats.activities.Cards: android.content.ComponentName startService(android.content.Intent)	INTENT	Negative flow.
overlay_android_samp	6	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	OTHER DATA	exts.whats.activities.Cards: android.content.ComponentName startService(android.content.Intent)	INTENT	Negative flow.
phospy	1	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	java.io.DataOutputStream: void writeUTF(java.lang.String)	NETWORK	This malicious flow reads device id and sends to a remote server via socket.
phospy	2	java.io.FileInputStream: void <init	FILE	java.io.DataOutputStream: void write(byte[],int,int)	NETWORK	This malicious flow reads JPG images from sd card and send to a remote server via socket
phospy	3	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	java.io.DataOutputStream: void write(byte[],int,int)	NETWORK	Negative flow.
phospy	4	java.io.FileInputStream: void <init	FILE	java.io.DataOutputStream: void writeUTF(java.lang.String)	NETWORK	Negative flow.
phospy	5	java.io.FileInputStream: void <init	FILE	java.io.DataOutputStream: void writeUTF(java.lang.String)	NETWORK	Negative flow.
proxy_samp	1	android.telephony.gsm.GsmCellLocation: int getLac()	LOCATION INFORMATION	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This flow logs location information.
proxy_samp	2	android.telephony.gsm.GsmCellLocation: int getLac()	LOCATION INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads location information and writes its into a private field, then sends it to a remote server per an HTTP POST request.
proxy_samp	3	java.io.File: void <init	FILE	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads a log file from external storage line by line and sends the content to a remote server. Each line of the file is written to an ArrayList. This ArrayList is used to construct the HTTP entity which will be sent to a remote server.
proxy_samp	4	android.accounts.AccountManager: android.accounts.Account[] getAccounts()	ACCOUNT INFORMATION	java.io.BufferedWriter: void write(java.lang.String)	FILE	This malicious flow reads account information and writes it to a log file.
proxy_samp	5	android.accounts.AccountManager: android.accounts.Account[] getAccounts()	ACCOUNT INFORMATION	java.io.BufferedWriter: void write(java.lang.String)	FILE	This malicious flow reads account information and writes gmail acount into a private field. The gmail acount name will be logged into a file.
proxy_samp	6	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.BufferedWriter: void write(java.lang.String)	FILE	This malicious flow reads sms from database and writes it a log file.
proxy_samp	7	android.net.wifi.WifiManager: android.net.wifi.WifiInfo getConnectionInfo()	NETWORK INFORMATION	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow reads wifi connection information and logs it.
proxy_samp	8	android.telephony.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow starts in a BroadcastReceiver. It reads sms message and logs it.
proxy_samp	9	android.accounts.AccountManager: android.accounts.Account[] getAccounts()	ACCOUNT INFORMATION	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs account information on the device.
proxy_samp	10	android.accounts.AccountManager: android.accounts.Account[] getAccounts()	ACCOUNT INFORMATION	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow sends Google account name to a remote server via a HTTP POST request.
proxy_samp	11	android.telephony.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	java.io.BufferedWriter: void write(java.lang.String)	FILE	This malicious flow reads incomming SMS message and writes to a log file.
proxy_samp	12	android.net.wifi.WifiManager: android.net.wifi.WifiInfo getConnectionInfo()	NETWORK INFORMATION	java.io.BufferedWriter: void write(java.lang.String)	FILE	This malicious flow reads WIFI information and writes to a log file.
proxy_samp	13	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs incoming SMS messages.
proxy_samp	14	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.BufferedWriter: void write(java.lang.String)	FILE	This malicious flow saves phone call logs to a log file when the ProxyService is destoryed. This file will be later uploaded to a remote server
proxy_samp	15	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.BufferedWriter: void write(java.lang.String)	FILE	This malicious flow saves phone call logs to a log file when the ProxyService is destroyed. This file will be later uploaded to a remote server.
proxy_samp	16	java.io.File: void <init	FILE	java.io.BufferedWriter: void write(java.lang.String)	FILE	Negative flow.
proxy_samp	17	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs phone calls.
proxy_samp	18	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	This malicious flow logs phone calls.
proxy_samp	19	android.net.wifi.WifiManager: android.net.wifi.WifiInfo getConnectionInfo()	NETWORK INFORMATION	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	Negative flow.
proxy_samp	20	java.io.File: void <init	FILE	android.util.Log: int i(java.lang.String,java.lang.String)	LOG	Negative flow.
remote_control_smack	1	android.location.LocationManager: android.location.Location getLastKnownLocation(java.lang.String)	LOCATION INFORMATION	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads location information (longitude) and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	2	android.location.LocationManager: android.location.Location getLastKnownLocation(java.lang.String)	LOCATION INFORMATION	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads location information (latitude) and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	3	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads SMS information (address) and saves it a text file which will be later uploaded to a remote server.

remote_control_smack	4	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads SMS information (message body) and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	5	android.database.Cursor: int getInt(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads SMS information (message type) and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	6	android.database.Cursor: long getLong(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads SMS information (date) and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	7	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious partial flow reads call record (number) and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	8	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious partial flow reads call record (name) and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	9	android.database.Cursor: int getInt(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads call record (type) and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	10	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads call record (date) and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	11	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads contact information and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	12	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads contact information and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	13	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads calender information and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	14	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads calender information and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	15	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads calender information and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	16	android.database.Cursor: long getLong(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads calender information and saves it a text file which will be later uploaded to a remote server.
remote_control_smack	17	android.database.Cursor: java.lang.String getString(int)	DATABASE	java.io.FileWriter: java.io.Writer append(java.lang.CharSequence)	FILE	This malicious flow reads calender information and saves it a text file which will be later uploaded to a remote server.
repane	1	java.io.File: void <init	FILE	android.content.Context: void startActivity(android.content.Intent)	INTENT	This malicious flows installs an APK that is shipped with the App.
roidsec	1	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.OutputStream: void write(byte[])	NETWORK	This malicious flow reads phone numbers from call logs and sends to a remote server via socket.
roidsec	2	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.OutputStream: void write(byte[])	NETWORK	This malicious flow reads phone numbers from contact and sends to aremote server via socket.
roidsec	3	android.content.pm.PackageManager: java.util.List getInstalledPackages(int)	SYSTEM SETTINGS	java.io.OutputStream: void write(byte[])	NETWORK	This malicious flow reads installed package name and sends to a remote server via socket.
roidsec	4	android.content.pm.PackageManager: java.util.List getInstalledPackages(int)	SYSTEM SETTINGS	java.io.OutputStream: void write(byte[])	NETWORK	This malicious flow reads information about installed apps and sends it to a remote server.
roidsec	5	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.OutputStream: void write(byte[])	NETWORK	This malicious flow reads incoming SMSs and sends them to a remote server.
roidsec	6	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.OutputStream: void write(byte[])	NETWORK	This malicious flow reads outgoing SMSs and sends them to a remote server.
samsapo	1	android.database.Cursor: java.lang.String getString(int)	DATABASE	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads SMS address and sends it to a remote server via HTTP Post request.
samsapo	2	android.telephony.gsm.SmsMessage: android.telephony.gsm.SmsMessage createFromPdu(byte[])	SMS MMS	com.android.tools.system.MyPostRequest: android.os.AsyncTask execute(java.lang.Object[])	NETWORK	This malicious flow reads incoming SMSs and sends them to a remote server via HTTP request.
samsapo	3	android.content.Context: java.lang.Object getSystemService(java.lang.String)	SYSTEM SETTINGS	java.lang.reflect.Method: java.lang.Object invoke(java.lang.Object,java.lang.Object[])	CRITICAL FUNCTION	This malicious flow uses reflection to terminate calls from user's contact.
samsapo	4	android.telephony.gsm.SmsMessage: android.telephony.gsm.SmsMessage createFromPdu(byte[])	SMS MMS	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow sends incoming SMS message to a remote server via a HTTP POST.
samsapo	5	android.content.Context: java.lang.Object getSystemService(java.lang.String)	SYSTEM SETTINGS	java.lang.reflect.Method: java.lang.Object invoke(java.lang.Object,java.lang.Object[])	CRITICAL FUNCTION	Negative flow.
save_me	1	android.widget.EditText: android.text.Editable getText()	ACCOUNT INFORMATION	android.content.ContentValues: void put(java.lang.String,java.lang.String)	DATABASE	This malicious flow reads user name from the input field and stores into database, which will be queried by the service called CO running in the background and sent to a remote server via HTTP requests.

save_me	2	android.widget.EditText: android.text.Editable getText()	ACCOUNT INFORMATION	android.content.ContentValues: void put(java.lang.String,java.lang.String)	DATABASE	This malicious flow reads phone number from the input field and stores into database, which will be queried by the service called CO running in the background and sent to a remote server via HTTP requests.
save_me	3	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This malicious flow reads the phone number from database and sends spam sms message to this phone number.
save_me	4	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads user name from database and puts into a ArrayList of name value pairs. This list is sent to a remote server per an HTTP Post request.
save_me	5	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads phone number from database and puts into a ArrayList of name value pairs. This list is sent to a remote server per an HTTP Post request.
save_me	6	android.net.wifi.WifiInfo: java.lang.String getMacAddress()	NETWORK INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads MAC address and writes it into a public field of the service CO. Value from this field is put into a ArrayList of name value pairs. This list is sent to a remote server per an HTTP Post request.
save_me	7	android.telephony.TelephonyManager: java.lang.String getSimCountryIso()	LOCATION INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads country information and writes it into a public field of the service CO. Value from this field is put into a ArrayList of name value pairs. This list is sent to a remote server per an HTTP Post request.
save_me	8	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads user name from database and puts into a ArrayList of name value pairs. This list is sent to a remote server per an HTTP Post request.
save_me	9	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads phone number from database and puts into a ArrayList of name value pairs. This list is sent to a remote server per an HTTP Post request.
save_me	10	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads MAC address from database and puts into a ArrayList of name value pairs. This list is sent to a remote server per an HTTP Post request.
save_me	11	android.widget.EditText: android.text.Editable getText()	ACCOUNT INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious ICC flow starts by reading phone number from the input field and stores the value into an intent. With this intent a new activity is stareted. Phone number is read from the intent, written into an ArrayList which is sent to a remote server via an HTTP post request.
save_me	12	android.widget.EditText: android.text.Editable getText()	ACCOUNT INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow sends user input to a remote server.
save_me	13	android.net.wifi.WifiInfo: java.lang.String getMacAddress()	NETWORK INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads MAC address and sends to a remote server.
save_me	14	android.net.wifi.WifiInfo: java.lang.String getMacAddress()	NETWORK INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads Mac address and sends it to a remote server.
save_me	15	android.net.wifi.WifiInfo: java.lang.String getMacAddress()	NETWORK INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads MAC address and sends it to a remote server.
save_me	16	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads user data from data base and sends it to a remote server.
save_me	17	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads user data from database and sends it a remote server.
save_me	18	android.telephony.TelephonyManager: java.lang.String getSimCountryIso()	LOCATION INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow send country info to a remote server.
save_me	19	android.telephony.TelephonyManager: java.lang.String getSimCountryIso()	LOCATION INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	Negative flow.
save_me	20	android.telephony.TelephonyManager: java.lang.String getSimCountryIso()	LOCATION INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	Negative flow.
save_me	21	android.net.wifi.WifiInfo: java.lang.String getMacAddress()	NETWORK INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow sends MAC address to a remote server.

save_me	22	android.net.wifi.WifiInfo: java.lang.String getMacAddress()	NETWORK INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads MAC address and sends it to a remote server.
save_me	23	android.net.wifi.WifiInfo: java.lang.String getMacAddress()	NETWORK INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads MAC address and sends it to a remote server.
save_me	24	android.net.wifi.WifiInfo: java.lang.String getMacAddress()	NETWORK INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads MAC address and sends it to a remote server.
save_me	25	android.net.wifi.WifiInfo: java.lang.String getMacAddress()	NETWORK INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads MAC address and sends it to a remote server.
save_me	26	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	android.content.ContentValues: void put(java.lang.String,java.lang.String)	DATABASE	Negative flow.
save_me	27	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	android.content.ContentValues: void put(java.lang.String,java.lang.String)	DATABASE	Negative flow.
save_me	28	android.database.sqlite.SQLiteDatabase: android.database.Cursor query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String,java.lang.String,java.lang.String)	DATABASE	android.content.ContentValues: void put(java.lang.String,java.lang.String)	DATABASE	Negative flow.
save_me	29	android.net.wifi.WifiInfo: java.lang.String getMacAddress()	NETWORK INFORMATION	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	Negative flow.
save_me	30	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	com.savemebeta.Scan: void startActivity(android.content.Intent)	INTENT	This malicious flow is a part of ICC flow which leaks contact information. The varaible var2 holds the contact name (see AddFriend class in com.savemebeta.Analyse.java).
save_me	31	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	CONTACT INFORMATION	com.savemebeta.Scan: void startActivity(android.content.Intent)	INTENT	This malicious flow is a part of ICC flow which leaks contact information. The varaible var2 holds the contact name (see AddFriend class in com.savemebeta.Analyse.java).
scipiex	1	android.telephony.TelephonyManager: java.lang.String getLine1Number()	UNIQUE IDENTIFIER	java.io.OutputStream: void write(byte[])	NETWORK	This malicious flow reads telephone number, stores it a static field and starts a service which listens to the incoming SMSs. The telephone number will be sent to a remote server when SMSs are received.
scipiex	2	android.telephony.SmsMessage: android.telephony.SmsMessage createFromPdu(byte[])	SMS MMS	java.io.OutputStream: void write(byte[])	NETWORK	This malicious flow reads received SMSs and send them to a remote server via a HTTP POST request.
scipiex	3	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.PrintWriter: void println(java.lang.String)	FILE	This malicious flow reads contacts and saves them to a file wich will be later sent to a remote server.
slocker_android_samp	1	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads device id and puts into a ArrayList of name value pairs. This list is sent to a remote server per an HTTP Post request.
slocker_android_samp	2	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads device id and puts into a ArrayList of name value pairs. This list is sent to a remote server per an HTTP Post request.
slocker_android_samp	3	android.widget.EditText: android.text.Editable getText()	ACCOUNT INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flows reads the inputed code and sends it to a remote server via HTTP POST request.
slocker_android_samp	4	android.widget.EditText: android.text.Editable getText()	ACCOUNT INFORMATION	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads the inputed code and sends to a remote server via HTTP POST request.
slocker_android_samp	5	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads device id and sends it to a remote server via an HTTP post.
smssend_packageInstaller	1	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	java.net.HttpURLConnection: java.io.OutputStream getOutputStream()	NETWORK	This malicious flow leaks the IMEI via an HTTP POST Request.
smssend_packageInstaller	2	android.telephony.SmsMessage: android.telephony.SmsMessage createFromPdu(byte[])	SMS MMS	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This is is an implicit flow. When a certain SMS is received it sends another SMS.
smssend_packageInstaller	3	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This malicious flow sends device id in a SMS message back to the sender when a certian SMS message is received.
smssend_packageInstaller	4	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This malicious flow sends a SMS contains the device id.

smssend_packageInstaller	5	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	java.net.HttpURLConnection: java.io.OutputStream getOutputStream()	NETWORK	This malicious flow sends device Id to a remote server.
smssilience_fake_vertu	1	android.telephony.TelephonyManager: java.lang.String getLine1Number()	UNIQUE IDENTIFIER	java.io.PrintWriter: void write(java.lang.String)	NETWORK	This malicious flow leaks the users phone number to a remote server.
smssilience_fake_vertu	2	android.telephony.SmsMessage: android.telephony.SmsMessage createFromPdu(byte[])	SMS MMS	java.io.PrintWriter: void write(java.lang.String)	NETWORK	This malicous flow sends the contents of the retrieved sms to a remote server.
smssilience_fake_vertu	3	android.telephony.TelephonyManager: java.lang.String getLine1Number()	UNIQUE IDENTIFIER	java.io.PrintWriter: void write(java.lang.String)	NETWORK	Negative flow.
smssilience_fake_vertu	4	android.telephony.TelephonyManager: java.lang.String getLine1Number()	UNIQUE IDENTIFIER	java.io.PrintWriter: void write(java.lang.String)	NETWORK	Negative flow.
smsstealer_kysn_assassincreed_android_samp	1	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads the types of sms messages and send to a remote server via an HTTP POST request.
smsstealer_kysn_assassincreed_android_samp	2	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads the addresses of sms messages and send to a remote server via an HTTP POST request.
smsstealer_kysn_assassincreed_android_samp	3	android.telephony.TelephonyManager: java.lang.String getSubscriberId()	UNIQUE IDENTIFIER	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads the IMSI of the device and sends it to a remote server via an HTTP POST request.
smsstealer_kysn_assassincreed_android_samp	4	java.net.HttpURLConnection: java.io.InputStream getInputStream()	INTERNET SOURCE	android.content.SharedPreferences$Editor: boolean commit()	OTHER STORAGE	This malicious flow reads command and decrypts information from a remote server and store data into shared perference.
smsstealer_kysn_assassincreed_android_samp	5	java.net.HttpURLConnection: java.io.InputStream getInputStream()	INTERNET SOURCE	android.telephony.gsm.SmsManager: void sendMultipartTextMessage(java.lang.String,java.lang.String,java.util.ArrayList,java.util.ArrayList,java.util.ArrayList)	SMS MMS	This malicious flow reads command and decrypts information from a remote server and sends a multi-part text based SMS.
sms_google	1	com.google.elements.Utils: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads the device id and puts it into a JSON object which will be sent to a remote sever via HTTP request.
sms_google	2	com.google.elements.Utils: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	org.apache.http.client.HttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow reads the device id and puts it into a JSON object which will be sent to a remote sever via HTTP request.
sms_google	3	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	SMS MMS	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This malicious partial flow listens to C&C command and sends SMS message in the background.
sms_google	4	android.content.Intent: java.lang.String getStringExtra(java.lang.String)	SMS MMS	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This malicious partical flow listens to C&C command and sends SMS message.
sms_send_locker_qqmagic	1	android.telephony.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This malicious flow exists in the onReceive callback method of a BroadcastReciver, it starts from reading a SMS message to a StringBuilder and ends by sending it via a text message to a strange number.
sms_send_locker_qqmagic	2	android.telephony.SmsMessage: java.lang.String getDisplayOriginatingAddress()	CONTACT INFORMATION	android.telephony.SmsManager: void sendTextMessage(java.lang.String,java.lang.String,java.lang.String,android.app.PendingIntent,android.app.PendingIntent)	SMS MMS	This malicious flow exists in the onReceive callback method of a BroadcastReciver, it starts from reading the SMS sender's address (phone number or email) and ends by sending it via a text message to a strange number.
sms_send_locker_qqmagic	3	android.telephony.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	android.content.Context: android.content.ComponentName startService(android.content.Intent)	INTENT	This malicious ICC flow cross one BroadcastReciver and one Service. This is only the partical flow in the BroadcastReciver. The flow starts in a broadcast receiver by reading SMS message to a StringBuilder, then uses this StringBuilder to construct an intent for a unknown service.
sms_send_locker_qqmagic	4	android.telephony.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	android.content.Context: android.content.ComponentName startService(android.content.Intent)	INTENT	Negative flow.
sms_send_locker_qqmagic	5	android.telephony.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	android.content.Context: android.content.ComponentName startService(android.content.Intent)	INTENT	
sms_send_locker_qqmagic	6	android.telephony.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	android.content.Context: android.content.ComponentName startService(android.content.Intent)	INTENT	This malicious ICC flow cross one BroadcastReciver and one Service. This is only the partical flow in the BroadcastReciver. The flow starts in a broadcast receiver by reading SMS message to a StringBuilder, then uses this StringBuilder to construct an intent for a unknown service.
sms_send_locker_qqmagic	7	android.telephony.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	android.content.Context: android.content.ComponentName startService(android.content.Intent)	INTENT	This malicious ICC flow cross one BroadcastReciver and one Service. This is only the partical flow in the BroadcastReciver. The flow starts in a broadcast receiver by reading SMS message to a StringBuilder, then uses this StringBuilder to construct an intent for a unknown service.

sms_send_locker_qqmagic	8	android.telephony.SmsMessage: java.lang.String getDisplayMessageBody()	SMS MMS	android.content.Context: android.content.ComponentName startService(android.content.Intent)	INTENT	This malicious ICC flow cross one BroadcastReciver and one Service. This is only the partical flow in the BroadcastReciver. The flow starts in a broadcast receiver by reading SMS message to a StringBuilder, then uses this StringBuilder to construct an intent for a unknown service.
stels_flashplayer_android_update	1	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.DataOutputStream: void write(byte[])	NETWORK	This malicious flow reads contacts and send them to a remote server via HTTP Post request.
stels_flashplayer_android_update	2	android.telephony.TelephonyManager: java.lang.String getSubscriberId()	UNIQUE IDENTIFIER	java.io.DataOutputStream: void write(byte[])	NETWORK	This malicious flow reads device's IMSI number and stores it to the public field of class Settings. The IMSI number is sent a remote server via a HTTP POST request.
stels_flashplayer_android_update	3	android.content.pm.PackageManager: java.util.List getInstalledPackages(int)	SYSTEM SETTINGS	java.io.DataOutputStream: void write(byte[])	NETWORK	This malicious leaks information about installed apks on the victim's device to a remote server.
tetus	1	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	java.net.HttpURLConnection: void connect()	NETWORK	This malicious flow reads the IMEI and writs to a static field. IMEI storeds in this static field is sent it to a remote server.
tetus	2	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	java.net.HttpURLConnection: void connect()	NETWORK	This malicious flow reads the IMEI and sends it to a server.
the_interview_movieshow	1	java.net.URL: java.io.InputStream openStream()	FILE	com.movieshow.down.Badaccents: void startActivity(android.content.Intent)	INTENT	This malicious flow loads an apk file to external storage and installs this apk.
threatjapan_uracto	1	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	java.io.PrintWriter: java.io.PrintWriter append(java.lang.CharSequence)	FILE	This partial flow writes mail data into a file.
threatjapan_uracto	2	java.io.File: void <init	FILE	org.apache.http.impl.client.DefaultHttpClient: org.apache.http.HttpResponse execute(org.apache.http.client.methods.HttpUriRequest)	NETWORK	This malicious flow sends a log file storing mail data to a remote server.
vibleaker_android_samp	1	android.os.Environment: java.io.File getExternalStorageDirectory()	FILE	org.springframework.web.client.RestTemplate: org.springframework.http.ResponseEntity exchange(java.lang.String,org.springframework.http.HttpMethod,org.springframework.http.HttpEntity,java.lang.Class,java.lang.Object[])	NETWORK	This malicious flow starts by getting the external storage path and uploads data from the Viber directories to a remote server per an HTTP POST request. There are checks along the flow for checking if the app Viber is installed.
vibleaker_android_samp	2	java.io.File: void <init	FILE	org.springframework.web.client.RestTemplate: org.springframework.http.ResponseEntity exchange(java.lang.String,org.springframework.http.HttpMethod,org.springframework.http.HttpEntity,java.lang.Class,java.lang.Object[])	NETWORK	This malicious flow reading videos from the Viber directories to a remote server per an HTTP POST request. There are checks along the flow for checking if the app Viber is installed.
vibleaker_android_samp	3	java.io.File: void <init	FILE	java.io.File: java.lang.String getName()	NETWORK	This malicious flow sends image to a remote server.
vibleaker_android_samp	4	java.io.File: void <init	FILE	java.io.File: java.lang.String getName()	NETWORK	This malicious flow sends images to a remote server.
xbot_android_samp	1	android.telephony.SmsMessage: android.telephony.SmsMessage createFromPdu(byte[])	SMS MMS	org.mozilla.javascript.Function: java.lang.Object call(org.mozilla.javascript.Context,org.mozilla.javascript.Scriptable,org.mozilla.javascript.Scriptable,java.lang.Object[])	CRITICAL FUNCTION	This malicious partial ICC flow starts by reading the Intent with which the SMSHandler BroadcasReceiver started. The intent contains SMS messagees, which will be sent to a remote server with some script call defined by the malware writer.
xbot_android_samp	2	android.telephony.TelephonyManager: java.lang.String getDeviceId()	UNIQUE IDENTIFIER	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	This malicious flow reads the device id and put it into an Array of String, the array is used as the return value of the method getTelephonyInfo() of the class xAPI. An object of xAPI is injected into the WebView by addJavascriptInterface. The method getTelephonyInfo() can be invoked via JavaScript code.
xbot_android_samp	3	android.content.ContentResolver: android.database.Cursor query(android.net.Uri,java.lang.String[],java.lang.String,java.lang.String[],java.lang.String)	DATABASE	android.webkit.WebView: void addJavascriptInterface(java.lang.Object,java.lang.String)	CRITICAL FUNCTION	This malicous reads contacts and store them into an ArrayList, this ArrayList is used as the return value of the method getContacts() of the class xAPI.An object of xAPI is injected into the WebView by addJavascriptInterface. The method getContacts() can be invoked via JavaScript code.