Google Play Protect

Triada

the past
the present
and the (hopefully not existing) future

Łukasz Siewierski (*@maldr0id*), Virus Bulletin 2018

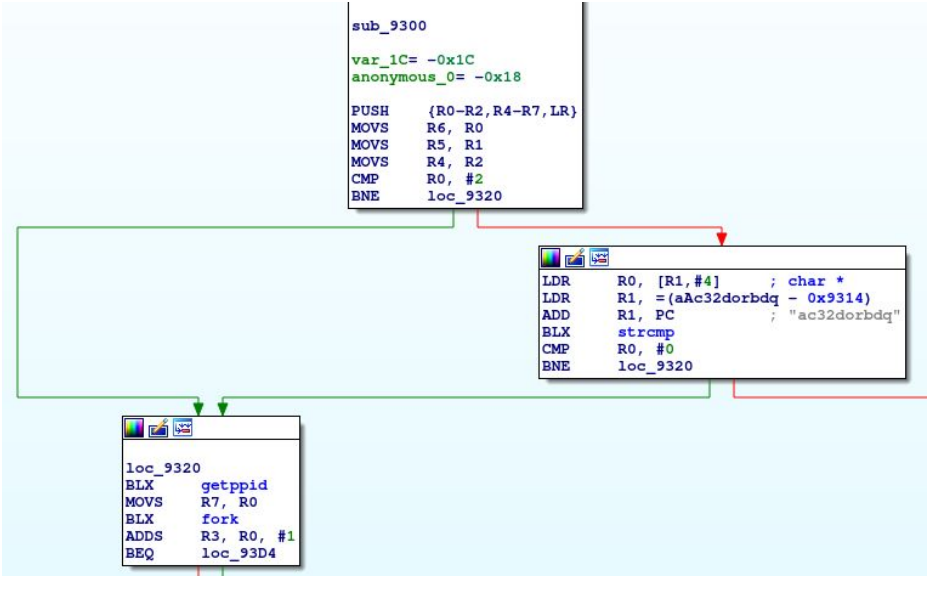
Triada evolution

1	[Old] rooting trojan	Early 2016
2	[New] system-level backdoor	July 2017
3	OEM outreach and system updates	

Historical information

Old Triada: rooting trojan

- First described by Kaspersky in March 2016
- Uses old exploits to gain root access on the device
- Drops a password protected su binary
- Library, component of the Triada infection chain, uses that binary to gain root and:
  - install apps in `/system`
  - turn off Google Play Protect
  - “clean” the device
  - make files immutable (chattr)
  - use OAuth tokens

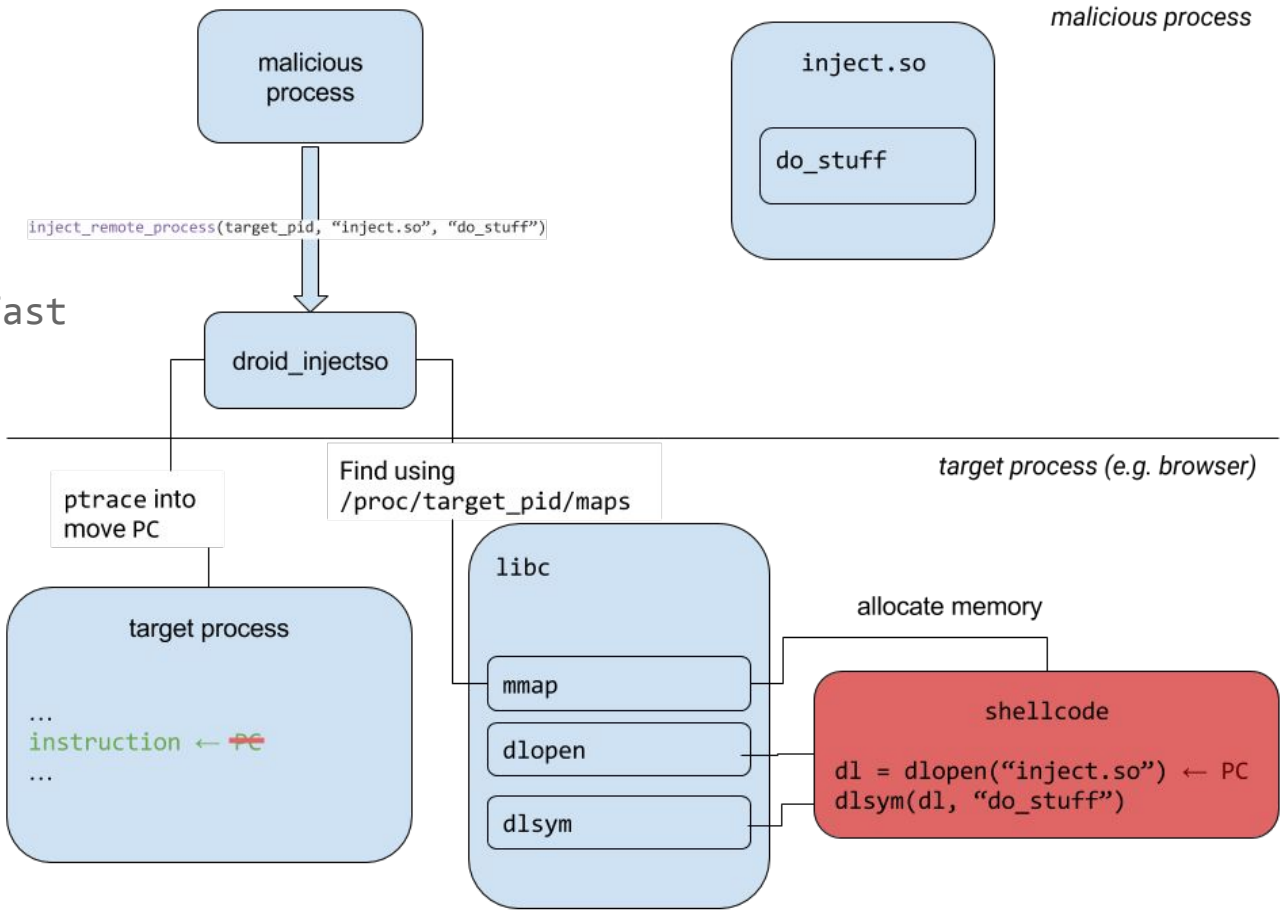


Google Play Protect

Old Triada: process and traffic hijacking

List of hijacked browsers:

- com.android.browser
- com.qihoo.browser
- com.ijinshan.browser_fast
- com.oupeng.browser



Google Play Protect

… and the double XOR

Two XORs with two different passwords…

For example Triada used this password pair for encryption:

googlefeeback

configopbinfo

```
public final byte[] a(byte[] p8) {
 int v0_5;
 int v1 = 0;
 if (p8 != null) {
  byte[] v2 = new byte[p8.length];
  int v0_2 = 0;
  while (v0_2 < p8.length) {
   v2[v0_2] = ((byte)(p8[v0_2] ^ this.a[(v0_2 %
this.a.length)]));
   v0_2++;
   }

  while (v1 < p8.length) {
   v2[v1] = ((byte)(v2[v1] ^ this.b[(v1 % this.b.length)]));
   v1++;
   }

  v0_5 = v2;
 } else {
  v0_5 = 0;
 }
  return v0_5;
 }
```

When rooting doesn’t work...

The line of code that started it all...

```
LABEL_13:
  v18 = -1;
LABEL_18:
  j___config_log_println(v7, v6, v10, v11, "cf89450001");
  if ( v10 )
```

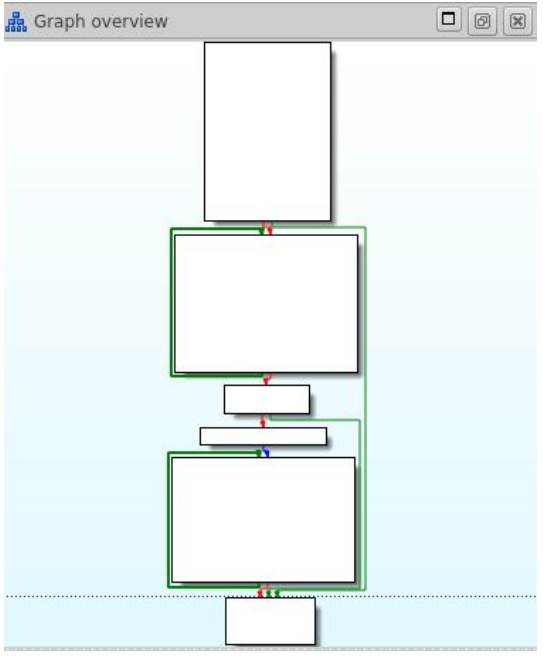
Triada backdoor added one additional line of code to the log function implementation in the Android framework.

Which means the backdoor code was executed in app context whenever log was called by the app.

This version was first described by Dr Web in July 2017.

… and a familiar code structure

.rodata:000…	00000012	C	is/pv~qop3s|mo(ce
.rodata:000…	00000015	C	is/pv~qop3s|mo(ddkv{
.rodata:000…	00000013	C	is/pv~qop3s|mo(dha
.rodata:000…	00000016	C	is/pv~qop3s|mo(surmed
.rodata:000…	00000017	C	is/pv~qop3s|mo(vdajyxh



The strings presented above are decrypted using the function on the right…

… which is a double XOR.

MMD file format and code injection

MMD is a file format used by Triada to inject code into other processes.

File naming scheme is:

`<md5 of the process name>36.jmd`

magic 8 bytes 4d 4d 44 00 0d 1a cb 00	hash length 4 bytes 20	decrypted file hash	file length 4 bytes	encrypted file

Google Play
Protect

Obfuscated code injections

`com.android.systemui (49b3d7ede8f829c04ca0dbc08dacb1ac):`

- To have the `GET_REAL_TASKS` permission

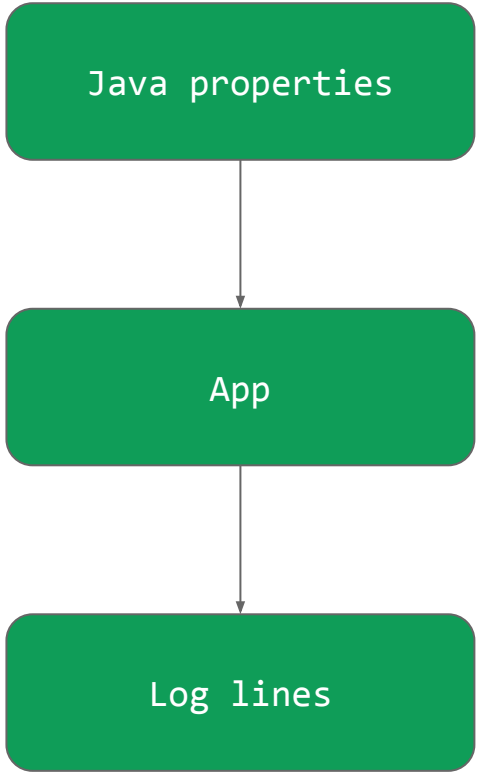
`com.android.vending (b5a5c5cb02ca09c784c5d88160e2ec24):`

1. 下载请求	1. download request
2. 下载结果	2. download result
3. 安装请求	3. install request (uses real, unpopular Google Play package names)
4. 安装结果	4. installation result
5. 激活请求	5. activation request
6. 激活结果	6. activation result
7. 拉活请求	7. pull request
8. 拉活结果	8. pull the results
9. 卸载请求	9. uninstall request
10. 卸载结果	10. uninstall result

Communication mechanisms

```
this.destPath = System.getProperty("os.config.ppgl.dir");
com.android.xb.tool.Lg.i(
    new StringBuilder().append("os.config.ppgl.dir=")
    .append(this.destPath).toString()
);
```

```
public static void entryLg(android.content.Context context)
{
    android.util.Log.i("COOLAPP", "coolapp_create");
    return;
}
```



WebView code modification

```
const-string/jumbo property, "persist.trd_yehuo_searchbox"
invoke-static {property},
Landroid/os/SystemProperties;->get(Ljava/lang/String;)Ljava/lang/String;
move-result-object property
const-string/jumbo true, "1"
invoke-virtual {property, property},
Ljava/lang/String;->equals(Ljava/lang/Object;)Z
move-result compare_result
if-eqz compare_result, :exit
invoke-static {url},
Landroid/webkit/WebView;->yhLoadUrl(Ljava/lang/String;)V
```

WebView code modification

```
const-string/jumbo property, "persist.trd_yehuo_searchbox"
invoke-static {property},
Landroid/os/SystemProperties;->get(Ljava/lang/String;)Ljava/lang/String;
move-result-object property
const-string/jumbo true, "1"
invoke-virtual {property, true},
Ljava/lang/String;->equals(Ljava/lang/Object;)Z
move-result compare_result
if-eqz compare_result, :exit
invoke-static {url},
Landroid/webkit/WebView;->yhLoadUrl(Ljava/lang/String;)V
```

The evolution of Triada versions

Fortunately Triada backdoor has version numbers.
The newest version that we know about is 1.5.1.

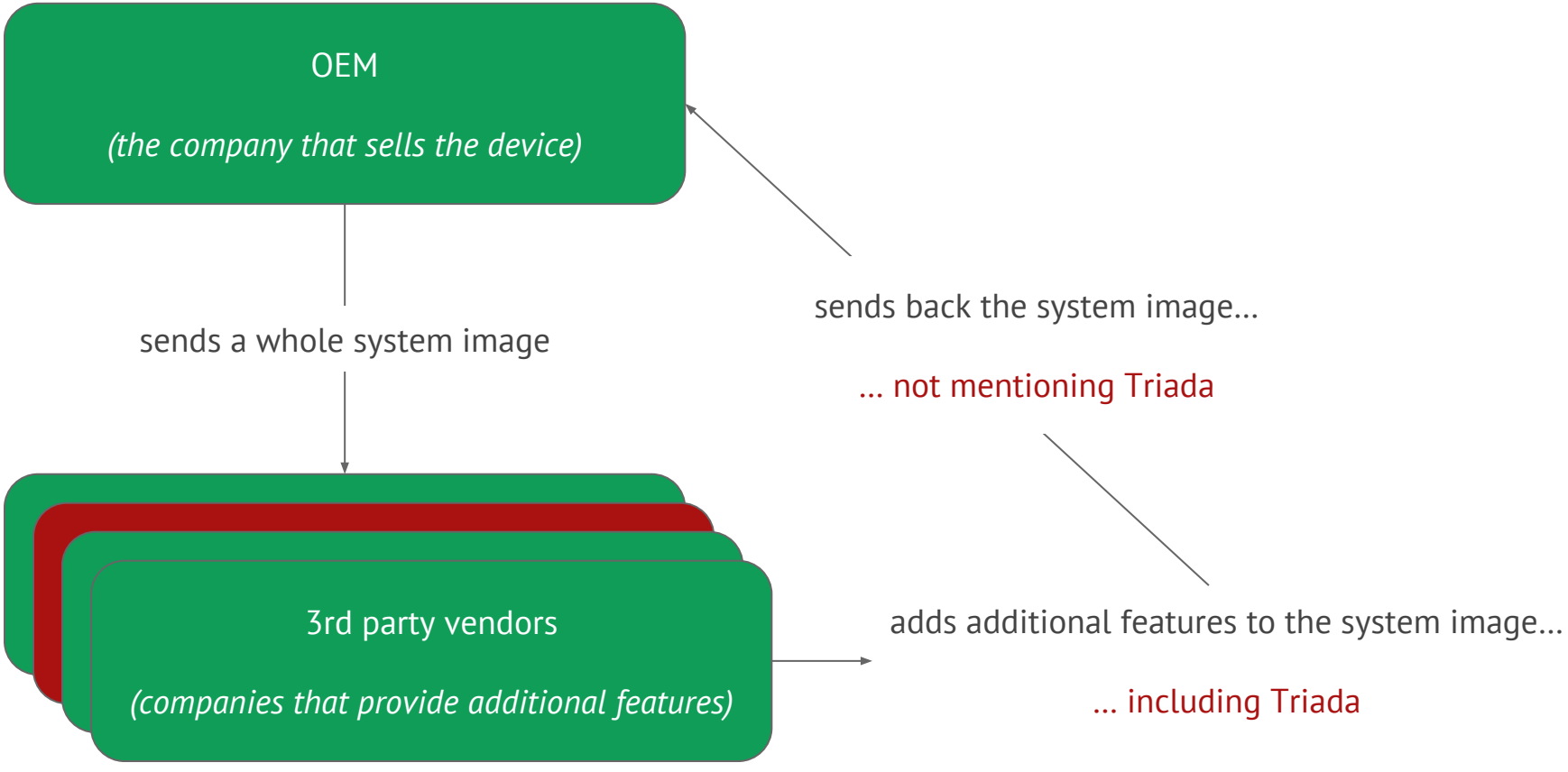
Changelog:

- MD5 hashes updated to SHA1
- Introduced a delay between backdoor actions
- Changed the code injection framework
- Code stubs pointing to a future possibility of backdooring systemui app
- Compatibility upgrades for Android Nougat

RE is done, time to protect the users!

Proprietary & Confidential Google Play

How did the backdoor get to the device?



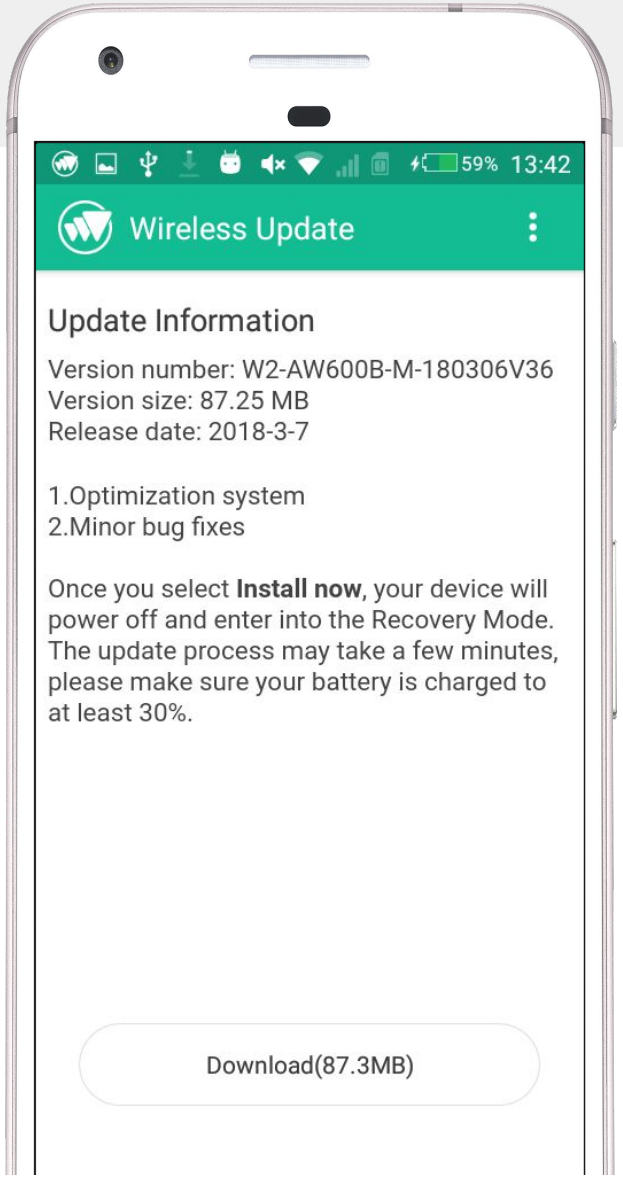
Google Play Protect

OEM outreach summary

We identified OEMs that had devices with Triada backdoor preinstalled on them.

We reached out to them and worked with them to remove the backdoor code.

OEMs started to remove Triada from devices by providing system updates, as seen here.



Google Play Protect

Summary

- Triada authors moved from a rooting trojan to a preinstalled backdoor
- Triada retained more or less the same functionality and purpose
- Reverse engineering shows multiple different code changes and injections on some Android 6 and 7 (Marshmallow and Nougat) devices
- We worked with all the affected OEMs and asked them to provide a system update that removes Triada
- System updates have been made available to the users

THANK YOU

Google Play
Protect