UNIVERSITY OF TORONTO

MUNK SCHOOL OF GLOBAL AFFAIRS

Join the Global Conversation

The Citizen Lab

Research Brief
August 2012

The SmartPhone Who Loved Me:

FinFisher Goes Mobile

Authors: Morgan Marquis-Boire, Bill Marzcak and Claudio Guarnieri

This post describes our work analyzing several samples which appear to be mobile variants of the FinFisher Toolkit, and ongoing scanning we are performing that has identified more apparent FinFisher command and control servers.

INTRODUCTION

Earlier this year, Bahraini Human Rights activists were targeted by an email campaign that delivered a sophisticated Trojan. In *From Bahrain with Love: FinFisher's Spy Kit Exposed?* we characterized the malware, and suggested that it appeared to be FinSpy, part of the FinFisher commercial surveillance toolkit. Vernon Silver concurrently reported our findings in Bloomberg, providing background on the attack and the analysis, and highlighting links to FinFisher's parent company, Gamma International.

After these initial reports, Rapid7, a Boston-based security company, produced a follow-up analysis that identified apparent FinFisher Command and Control (C&C) servers on five continents. After the release of the Rapid7 report, Gamma International representatives spoke with Bloomberg and The New York Times' Bits Blog, and denied that the servers found in 10 countries were instances of their products.

Following these analyses, we were contacted by both the security and activist communities with potentially interesting samples. From these, we identified several apparent mobile Trojans for the iOS, Android, BlackBerry, Windows Mobile and Symbian platforms. **Based on our analysis, we found these tools to be consistent in functionality with claims made in the documentation for the FinSpy Mobile product**, a component of the FinFisher toolkit. Several samples appear to be either demo versions or "unpackaged" versions ready to be customized, while others appear to be samples in active use.

Promotional literature describes this product as providing:

- Recording of common communications like Voice Calls, SMS/MMS and Emails
- Live Surveillance through silent calls
- File Download (Contacts, Calendar, Pictures, Files)
- Country Tracing of Target (GPS and Cell ID)
- Full Recording of all BlackBerry Messenger communications
- Covert Communications with Headquarters

In addition to analysis of these samples, we are conducting an ongoing scan for FinFisher C&C servers, and have identified potential servers in the following countries: **Bahrain, Brunei, the Czech Republic, Ethiopia, Indonesia, Mongolia, Singapore, the Netherlands, Turkmenistan, and the United Arab Emirates (UAE)**.

MOBILE TROJANS

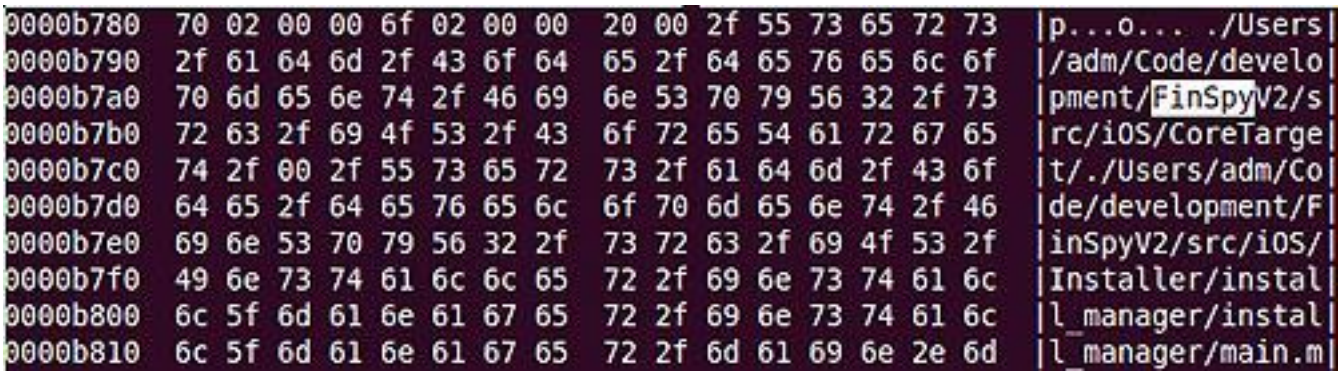
iOS

It was developed for Arm7, built against iOS SDK 5.1 on OSX 10.7.3 and it appears that it will run on iPhone 4, 4S, iPad 1, 2, 3, and iPod touch 3, 4 on iOS 4.0 and up.

The bundle is called “install_manager.app” and the contents of it are:

```
99621a7301bfd00d98c222a89900aeef ./data
1f73ebf8be52aa14d4d4546fb3242728 ./_CodeSignature/CodeResources
9273880e5baa5ac810f312f8bd29bd3f ./embedded.mobileprovision
2cbe06c89dc5a43ea0e0600ed496803e ./install_manager
23b7d7d024abb0f558420e098800bf27 ./PkgInfo
11e4821d845f369b610c31592f4316d9 ./Info.plist
ce7f5b3d4bfc7b4b0da6a06dccc515f2 ./en.lproj/InfoPlist.strings
3fa32da3b25862ba16af040be3451922 ./ResourceRules.plist
```

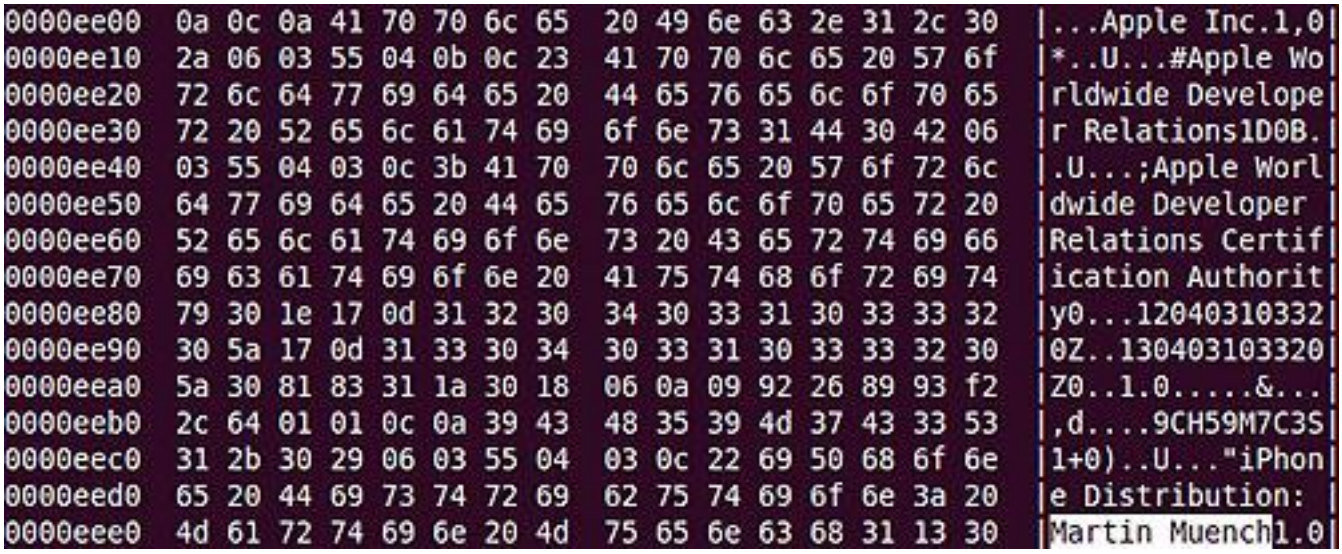
Investigation of the Mach-0 binary ‘install_manager’ reveals the text “FinSpy”:



Further references to “FinSpy” were identified in the binary:

/Users/adm/Code/development/FinSpyV2/src/iOS/CoreTarget/
/Users/adm/Code/development/FinSpyV2/src/iOS/Installer/install_manager/install_manager/main.m
/Users/adm/Code/development/FinSpyV2/src/iOS/Installer/install_manager/install_manager/zip/ioapi.c
/Users/adm/Code/development/FinSpyV2/src/iOS/Installer/install_manager/install_manager/zip/unzip.c
/Users/adm/Code/development/FinSpyV2/src/iOS/Installer/install_manager/install_manager/zip/crypt.h
/Users/adm/Code/development/FinSpyV2/src/iOS/Installer/install_manager/install_manager/zip/zip.c
/Users/adm/Code/development/FinSpyV2/src/iOS/Installer/install_manager/install_manager/zip/ZipArchive.mm
/Users/adm/Code/development/FinSpyV2/src/iOS/Installer/install_manager/install_manager/../../../CoreTarget/CoreTarget/GIFileOps.mm
/Users/adm/Code/development/FinSpyV2/src/iOS/Installer/install_manager/install_manager/../../../CoreTarget/CoreTarget/GIFileOps+Zip.m
/Users/adm/Code/development/FinSpyV2/src/iOS/Installer/install_manager/install_manager/../../../CoreTarget/CoreTarget/GIPath.mm

Additionally, it appears that a developer’s certificate belonging to Martin Muench, who is described in The New York Times as Managing Director of Gamma International GmbH and head of the FinFisher product portfolio, is used:



```
0000ee00  0a 0c 0a 41 70 70 6c 65  20 49 6e 63 2e 31 2c 30  |...Apple Inc.1,0|
0000ee10  2a 06 03 55 04 0b 0c 23  41 70 70 6c 65 20 57 6f  |*..U...#Apple Wo|
0000ee20  72 6c 64 77 69 64 65 20  44 65 76 65 6c 6f 70 65  |rldwide Develope|
0000ee30  72 20 52 65 6c 61 74 69  6f 6e 73 31 44 30 42 06  |r Relations1D0B.|
0000ee40  03 55 04 03 0c 3b 41 70  70 6c 65 20 57 6f 72 6c  |.U...;Apple Worl|
0000ee50  64 77 69 64 65 20 44 65  76 65 6c 6f 70 65 72 20  |dwide Developer |
0000ee60  52 65 6c 61 74 69 6f 6e  73 20 43 65 72 74 69 66  |Relations Certif|
0000ee70  69 63 61 74 69 6f 6e 20  41 75 74 68 6f 72 69 74  |ication Authorit|
0000ee80  79 30 1e 17 0d 31 32 30  34 30 33 31 30 33 33 32  |y0...12040310332|
0000ee90  30 5a 17 0d 31 33 30 34  30 33 31 30 33 33 32 30  |0Z..130403103320|
0000eea0  5a 30 81 83 31 1a 30 18  06 0a 09 92 26 89 93 f2  |Z0..1.0.....&...|
0000eeb0  2c 64 01 01 0c 0a 39 43  48 35 39 4d 37 43 33 53  |,d....9CH59M7C3S|
0000eec0  31 2b 30 29 06 03 55 04  03 0c 22 69 50 68 6f 6e  |1+0)..U..."iPhon|
0000eed0  65 20 44 69 73 74 72 69  62 75 74 69 6f 6e 3a 20  |e Distribution: |
0000eee0  4d 61 72 74 69 6e 20 4d  75 65 6e 63 68 31 13 30  |Martin Muench1.0|
```

An ad-hoc distribution profile is present: “testapp”:

UUID: “E0A4FAD7-E414-4F39-9DB3-5A845D5124BC”.
Will expire on 02.04.2013.
The profile matches the bundle ID (home.install-manager).
The profile was signed by 3 certificates.
The profile may be used by one developer:
Developer Certificate “iPhone Distribution: Martin Muench”.
This certificate was used to sign the bundle

The code signature contains 3 certificates:

Certificate “Apple Root CA”:
Will expire on 09.02.2035.
Your keychain contains this root certificate.
Certificate “Apple Worldwide Developer Relations Certification Authority”:
Will expire on 14.02.2016.
Certificate “iPhone Distribution: Martin Muench”:
Will expire on 03.04.2013.
SHA1 fingerprint: “1F921F276754ED8441D99FB0222A096A0B6E5C65”.

The Application has been provisioned to run on the following devices, represented here by their Unique Device Identifiers (UDID):

31b4f49bc9007f98b55df555b107cba841219a21,
73b94de27cb5841ff387078c175238d6abac44b2,
0b47179108f7ad5462ed386bc59520da8bfcea86,
320184fb96154522e6a7bd86dcd0c7a9805ce7c0,
11432945ee0b84c7b72e293cbe9acef48f900628,
5a3df0593f1b39b61e3c180f34b9682429f21b4f,
b5bfa7db6a0781827241901d6b67b9d4e5d5dce8

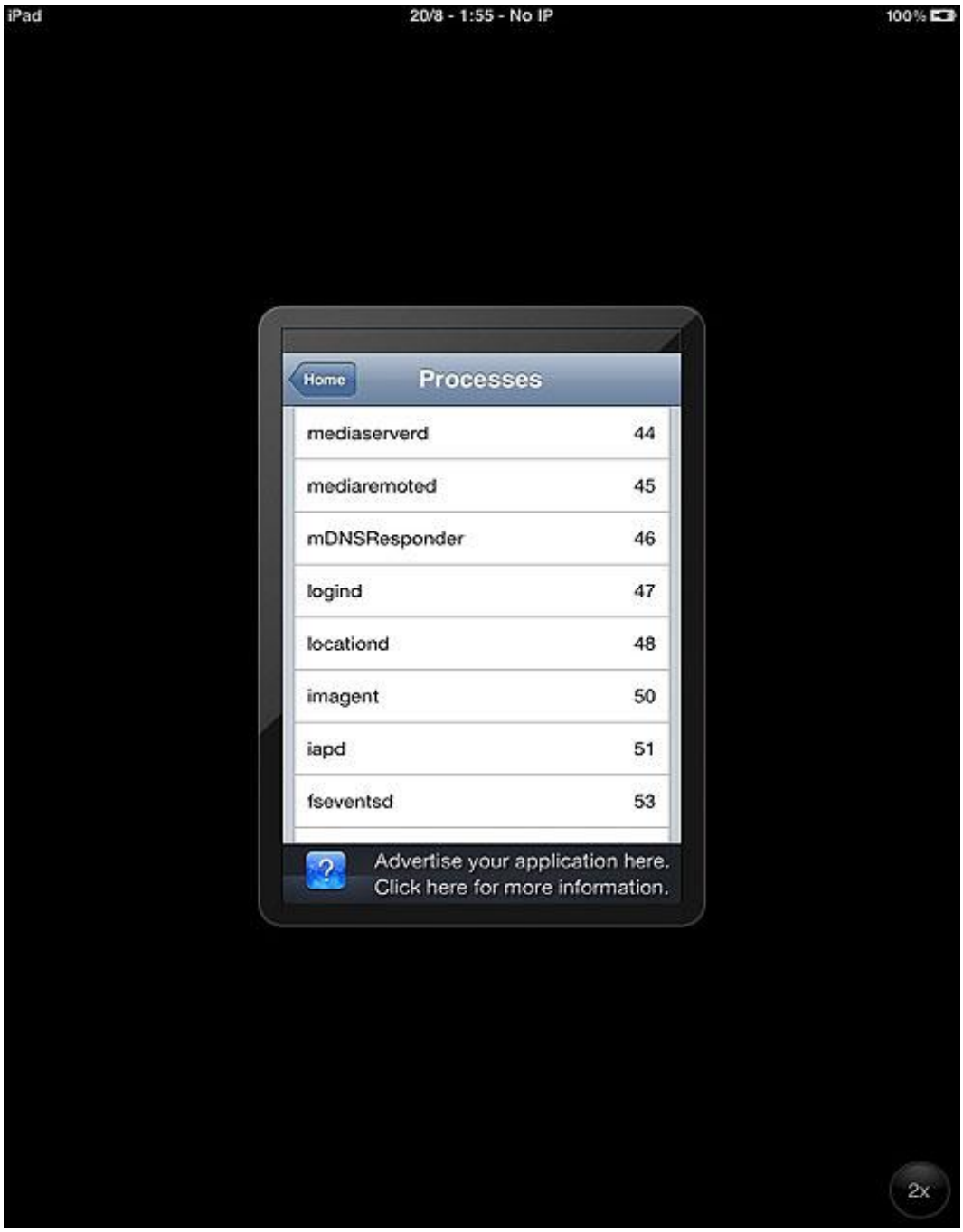
The file is hidden using Spring Board options, and on execution the sample writes out logind.app to /System/Library/CoreServices. 'logind' exists on OSX but not normally on iOS.

It then installs: /System/Library/LaunchDaemons/com.apple.logind.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
      <key>Disabled</key>
      <false/>
      <key>Label</key>
      <string>home.logind</string>
      <key>OnDemand</key>
      <false/>
      <key>ProgramArguments</key>
      <array>
            <string>/System/Library/CoreServices/logind.app/logind</string>
            <string></string>
            <string></string>
      </array>
      <key>StandardErrorPath</key>
      <string>/dev/null</string>
</dict>
</plist>
```

This creates persistence on reboot. It launches the logind process, then deletes install_manager.app.

On reboot it runs early in the boot process with ID 47:



This then drops SyncData.app. This application is signed, and the provisioning stipulates:

"Reliance on this certificate by any party assumes acceptance of the then applicable standard terms and conditions of use, certificate policy and certification practice statements."

Further legal analysis would be necessary to determine whether the program violated the terms of use at the time of its creation.

This application appears to provide functionality for call logging:

/Users/adm/Code/development/FinSpyV2/src/iOS/CoreTarget/CoreTarget/MobileLoggingDataTLV.m
_OBJC_METACLASS_$_MobileLoggingDataTLV
_OBJC_CLASS_$_MobileLoggingDataTLV

Exfiltration of contacts:

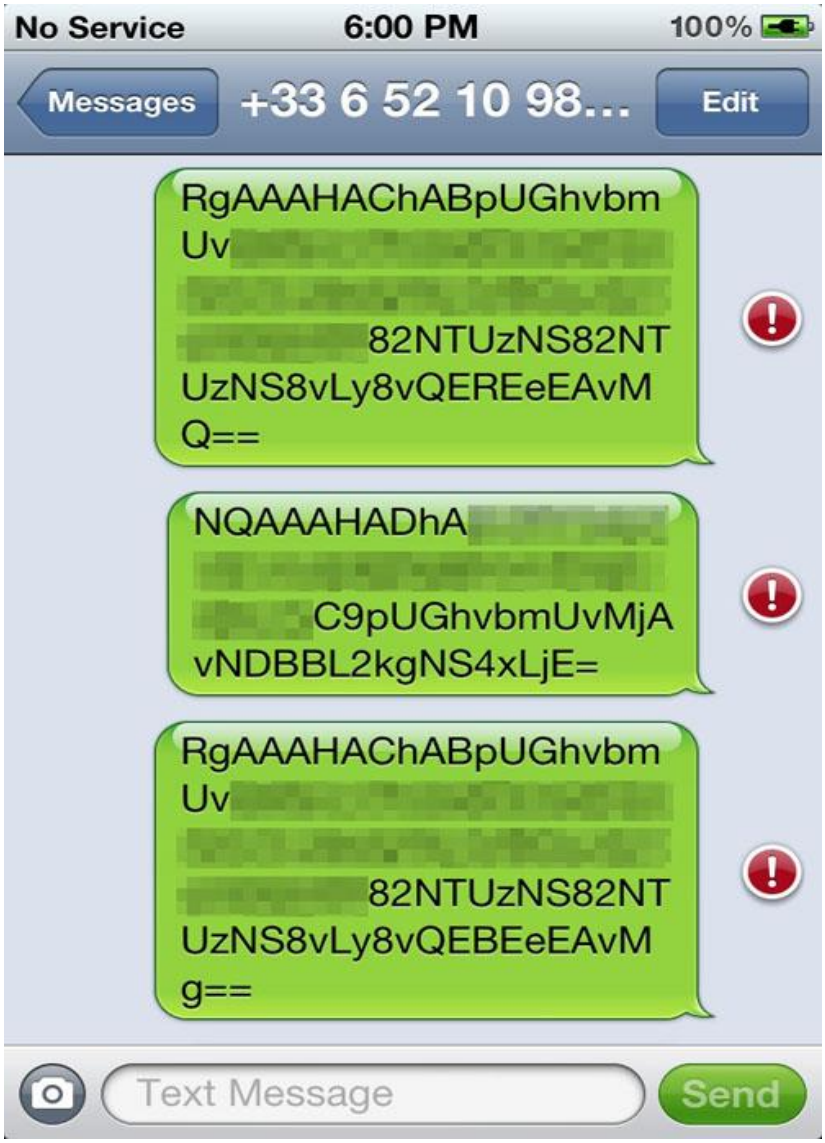
/Users/adm/Code/development/FinSpyV2/src/iOS/CoreTarget/CoreTarget/GIAddressBookModule.m
/Users/adm/Library/Developer/Xcode/DerivedData/CoreTarget-gqciilooqcckafgxlngvjezpbymr/Build/Intermediates/CoreTarget.build/Release-iphoneos/SyncData.build/Objects-normal/armv7/GIAddressBookModule.o
-[XXXVIII_cI getAddresses:]
/Users/adm/Code/development/FinSpyV2/src/iOS/CoreTarget/CoreTarget/GIAddressBookModuleData.m

Target location enumeration:

@_OBJC_CLASS_$_CLLocationManager
/Users/adm/Code/development/FinSpyV2/src/iOS/CoreTarget/CoreTarget/GILocationManager.m
/Users/adm/Library/Developer/Xcode/DerivedData/CoreTarget-gqciilooqcckafgxlngvjezpbymr/Build/Intermediates/CoreTarget.build/Release-iphoneos/SyncData.build/Objects-normal/armv7/GILocationManager.o

As well as arbitrary data exfiltration, SMS interception and more.

SyncData.app exfiltrates base64 encoded data about the device (including the IMEI, IMSI etc) to a remote cellular number.



The 'logind' process attempts to talk to a remote command and control server, the configuration information for which appears to be stored in base64 encoded form in "SyncData.app/84C.dat".

The _CodeSignature/CodeResources file suggests that install manager drops logind.app, SyncData.app and Trampoline.app (Trampoline.app has not been examined).

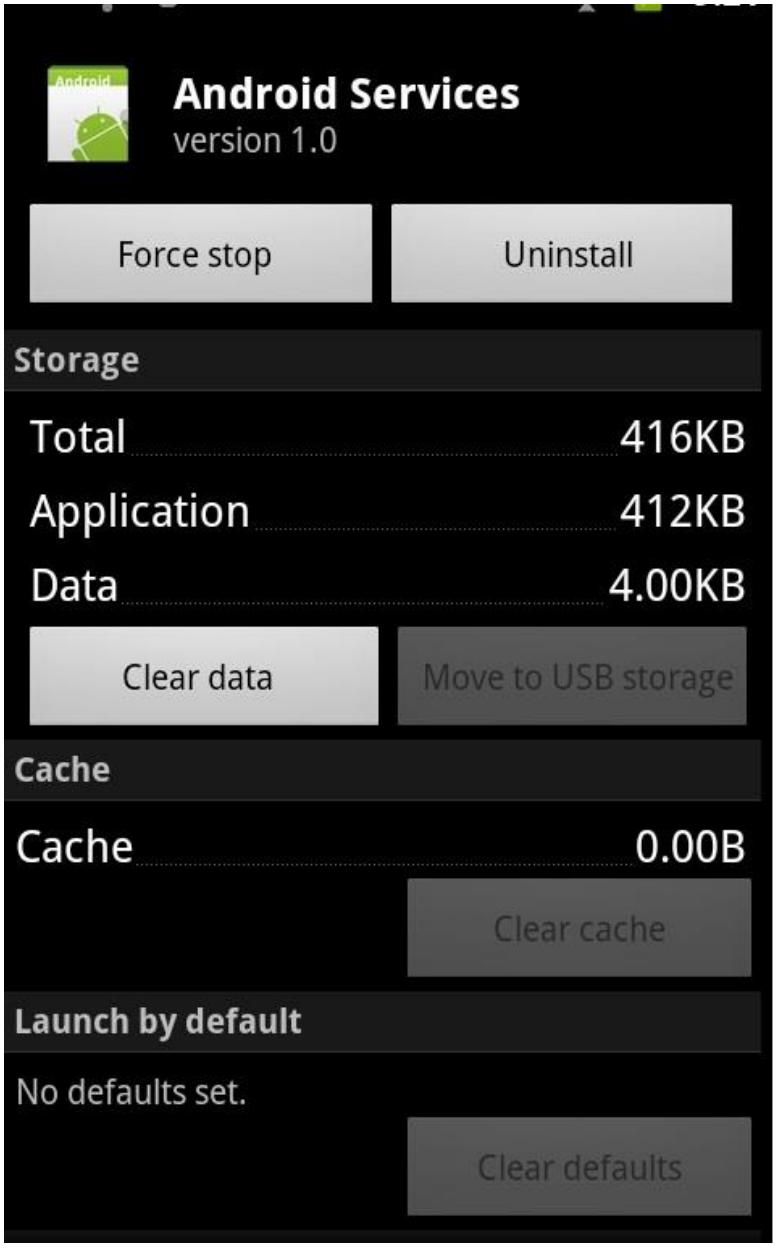
org.logind.ctp.archive/logind.app/logind
org.logind.ctp.archive/SyncData.app/SyncData
org.logind.ctp.archive/trampoline.app/trampoline

Android

The Android samples identified come in the form of APKs.

2e96e343ac10f5d9ace680e456c083e4eceb108f7209aa1e849f11a239e7a682 0d798ca0b2d0ea9bad251125973d8800ad3043e51d4cc6d0d57b971a97d3af2d 72a522d0d3dcd0dc026b02ab9535e87a9f5664bc5587fd33bb4a48094bce0537

The application appears to install itself as “Android Services”:



It requests the following permissions:

```
android.permission.ACCESS_COARSE_LOCATION
android.permission.ACCESS_FINE_LOCATION
android.permission.INTERNET
android.permission.READ_PHONE_STATE
android.permission.ACCESS_NETWORK_STATE
android.permission.READ_CONTACTS
android.permission.READ_SMS
android.permission.SEND_SMS
android.permission.RECEIVE_SMS
android.permission.WRITE_SMS
android.permission.RECEIVE_MMS
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.PROCESS_OUTGOING_CALLS
android.permission.ACCESS_NETWORK_STATE
android.permission.ACCESS_WIFI_STATE
android.permission.WAKE_LOCK
android.permission.CHANGE_WIFI_STATE
android.permission.MODIFY_PHONE_STATE
android.permission.BLUETOOTH
android.permission.RECEIVE_WAP_PUSH
```

The first 200 files in the apk are named "assets/Configurations/dummsX.dat", where X is a number from 0-199. The files are 0 bytes in length. The file header entries in the compressed file are normal, but the directory header entries contain configuration information.

The code in the my.api.Extractor.getConfiguration() method opens up the APK file and searches for directory entry headers (PK\x01\x02) then copies 6 bytes from the entry starting at offset 36. These are the "internal file attributes" and "external file attributes" fields. The code grabs these sequences until it hits a 0 value.This creates a base64 encoded string.

The app decodes this string and stores it in a file named 84c.dat (similar to the iOS sample discussed earlier).

Here's the output from one of the samples:

```
KQIAAJBb/gAhAgAAoDOEAAwAAABQE/4AAAAAABAAAABgV/4AAAAAAAAAAAAMAAAAQ
BX+AAAAAAAPAAAAcFj+AG1qbV9BTkQMAAAAQGGEACwBAAANAAAAkGSEAIKHhoGDJgA
AAHA3gABkZW1vLWRlLmdhbW1hLWludGVybmF0aW9uYWwuZGUbAAAAcDeAAGZmLWRlbW8
uYmxvZ2Rucy5vcmcMAAAAQDiAAFAAAAAMAAAAQDiAAFcEAAAMAAAAQDiAAFgEAAAVA
AAAcGOEACs0OTE3MjY2NTM4MDAWAAAAcGqEACs0OTg5NTQ5OTg5OTA4DwAAAHBmhABt
am1fQU5EDAAAAEBlhACmNqEPDAAAAEAh/gAoBAAADAAAAEANgAB7AAAADAAAAEBohA
AAAAAADAAAAEA7gAAAAAACgAAAJBghACtEAoAAACQYoQAwAAJAAAAsGeEAAAIAAA
AkMZxAIwAAACQeYQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEBAQEAAQEAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAPQAAAJA0RQA1AAAAoDNFAAwAAABAQUUA6AMAAAwAAABAQEUAL
AEAAAkAAAAwQkUAAAwAAACQZIQAh4aFgQ==
```

The Base64 decoded hexdump is:

```
00000000  29 02 00 00 90 5b fe 00  21 02 00 00 a0 33 84 00  |)....[..!....3..|
00000010  0c 00 00 00 50 13 fe 00  00 00 00 00 10 00 00 00  |....P...........|
00000020  60 57 fe 00 00 00 00 00  00 00 00 00 0c 00 00 00  |`W..............|
00000030  40 15 fe 00 00 00 00 00  0f 00 00 00 70 58 fe 00  |@...........pX..|
00000040  6d 6a 6d 5f 41 4e 44 0c  00 00 00 40 61 84 00 2c  |mjm_AND....@a..,|
00000050  01 00 00 0d 00 00 00 90  64 84 00 82 87 86 81 83  |........d.......|
00000060  26 00 00 00 70 37 80 00  64 65 6d 6f 2d 64 65 2e  |&...p7..demo-de.|
00000070  67 61 6d 6d 61 2d 69 6e  74 65 72 6e 61 74 69 6f  |gamma-internatio|
00000080  6e 61 6c 2e 64 65 1b 00  00 00 70 37 80 00 66 66  |nal.de....p7..ff|
00000090  2d 64 65 6d 6f 2e 62 6c  6f 67 64 6e 73 2e 6f 72  |-demo.blogdns.or|
000000a0  67 0c 00 00 00 40 38 80  00 50 00 00 00 0c 00 00  |g....@8..P......|
000000b0  00 40 38 80 00 57 04 00  00 0c 00 00 00 40 38 80  |.@8..W.......@8.|
000000c0  00 58 04 00 00 15 00 00  00 70 63 84 00 2b 34 39  |.X.......pc..+49|
000000d0  31 37 32 36 36 35 33 38  30 30 16 00 00 00 70 6a  |1726653800....pj|
000000e0  84 00 2b 34 39 38 39 35  34 39 39 38 39 39 30 38  |..+4989549989908|
000000f0  0f 00 00 00 70 66 84 00  6d 6a 6d 5f 41 4e 44 0c  |....pf..mjm_AND.|
00000100  00 00 00 40 65 84 00 a6  36 a1 0f 0c 00 00 00 40  |...@e...6......@|
00000110  21 fe 00 28 04 00 00 0c  00 00 00 40 0d 80 00 7b  |!..(.......@...{|
00000120  00 00 00 0c 00 00 00 40  68 84 00 00 00 00 00 0c  |.......@h.......|
00000130  00 00 00 40 3b 80 00 00  00 00 00 0a 00 00 00 90  |...@;...........|
00000140  60 84 00 ad 10 0a 00 00  00 90 62 84 00 c0 00 09  |`.........b.....|
00000150  00 00 00 b0 67 84 00 00  08 00 00 00 90 c6 71 00  |....g.........q.|
00000160  8c 00 00 00 90 79 84 00  00 00 00 00 00 00 00 00  |.....y..........|
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

Note that the hostnames demo-de.gamma-international.de and ff-demo.blogdns.org are suggestive of a demo or pre-customisation version of the FinSpy Mobile tool and are similar to domains identified in our previous report.

We identified samples structurally similar to this sample that spoke to servers in the **United Kingdom** and the **Czech Republic**:

Sample: 2e96e343ac10f5d9ace680e456c083e4eceb108f7209aa1e849f11a239e7a682
Command and Control: 80.95.253.44
Country: Czech Republic
Company: T-Systems Czech Republic

Sample: 0d798ca0b2d0ea9bad251125973d8800ad3043e51d4cc6d0d57b971a97d3af2d
Command and Control: 212.56.102.38
Country: United Kingdom
Company: PlusNet Technologies

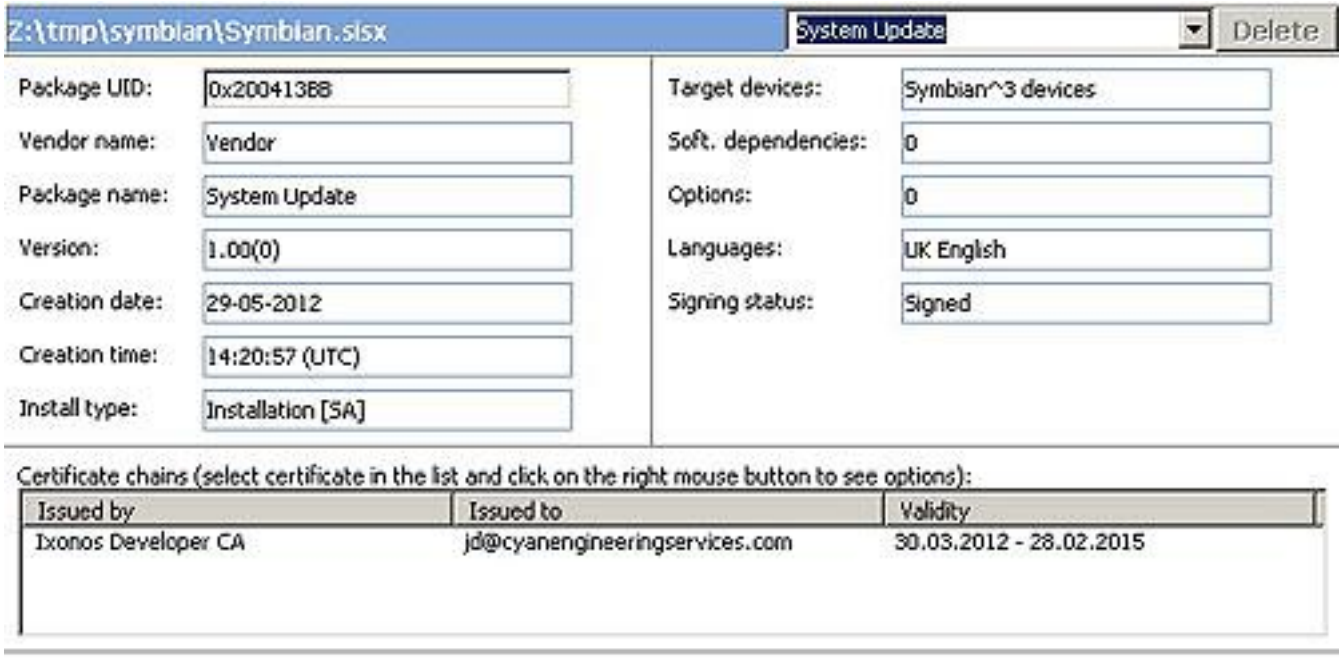
Note that the Czech sample speaks to the same command and control server previously identified by Rapid7.

Symbian

Samples for Nokia's Symbian platform were identified:

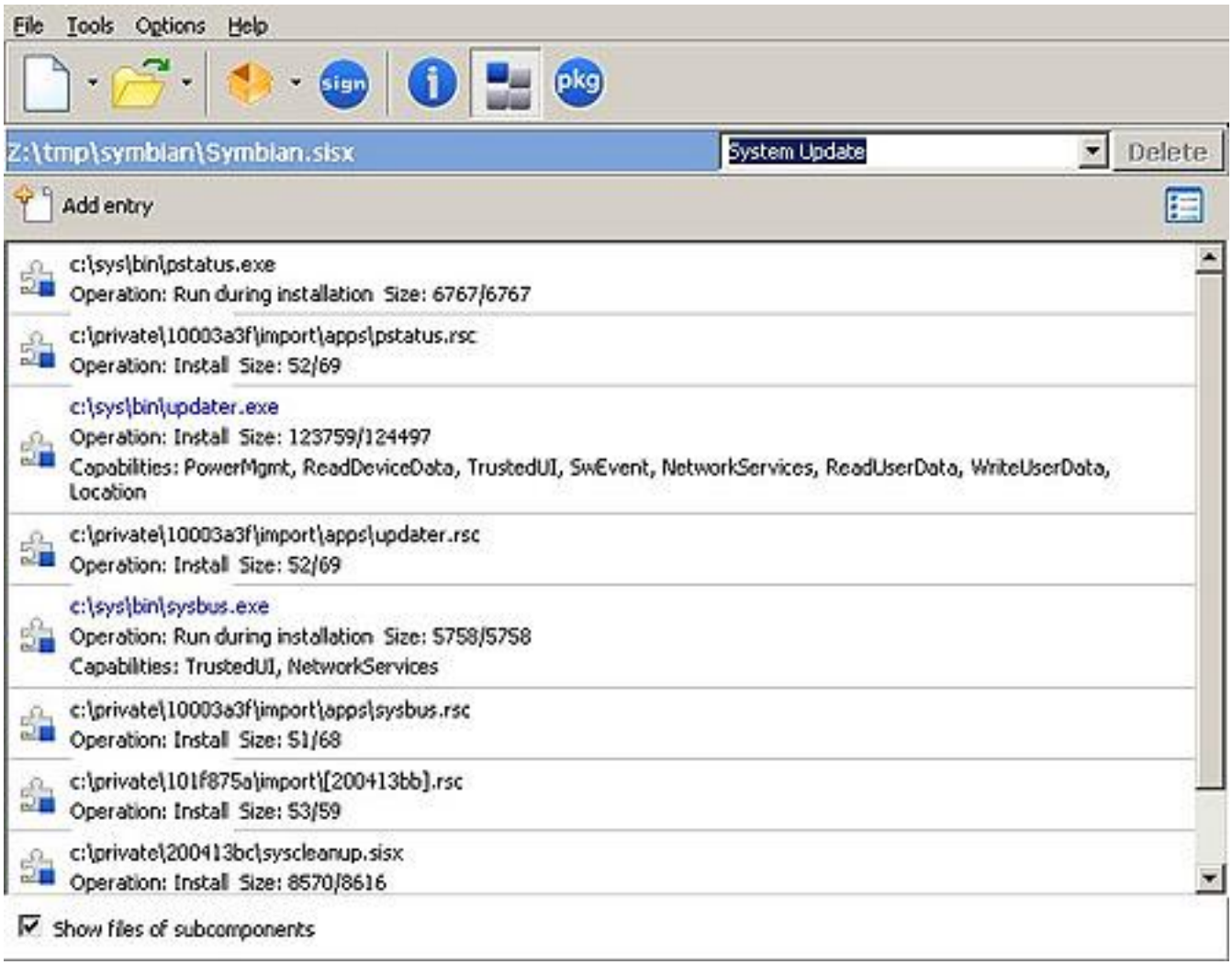
1e7e53b0d5fabcf12cd1bed4bd9ac561a3f4f6f8a8ddc5d1f3d2f3e2e9da0116 Symbian.sisx
eee80733f9664384d6bac4d4e27304748af9ee158d3c2987af5879ef83a59da0 mysym.sisx

The first sample ("Symbian.sisx") identifies itself as "System Update" and appears to have been built on the 29th of May 2012, at 14:20:57 UTC.



The certificate is registered to a jd@cyanengineeringservices.com. WHOIS information indicates that www.cyanengineeringservices.com was anonymously registered (date of first registration: 07-Mar-07) with GoDaddy using Domains By Proxy. Although it includes an attractive front page that states "Mobile Software Development" for "Windows Mobile, iPhone, Android, Symbian and Blackberry," all links (e.g. "Products" "About Us" or "Contacts") lead to an "under construction" blank page.

The sample contains the following components:



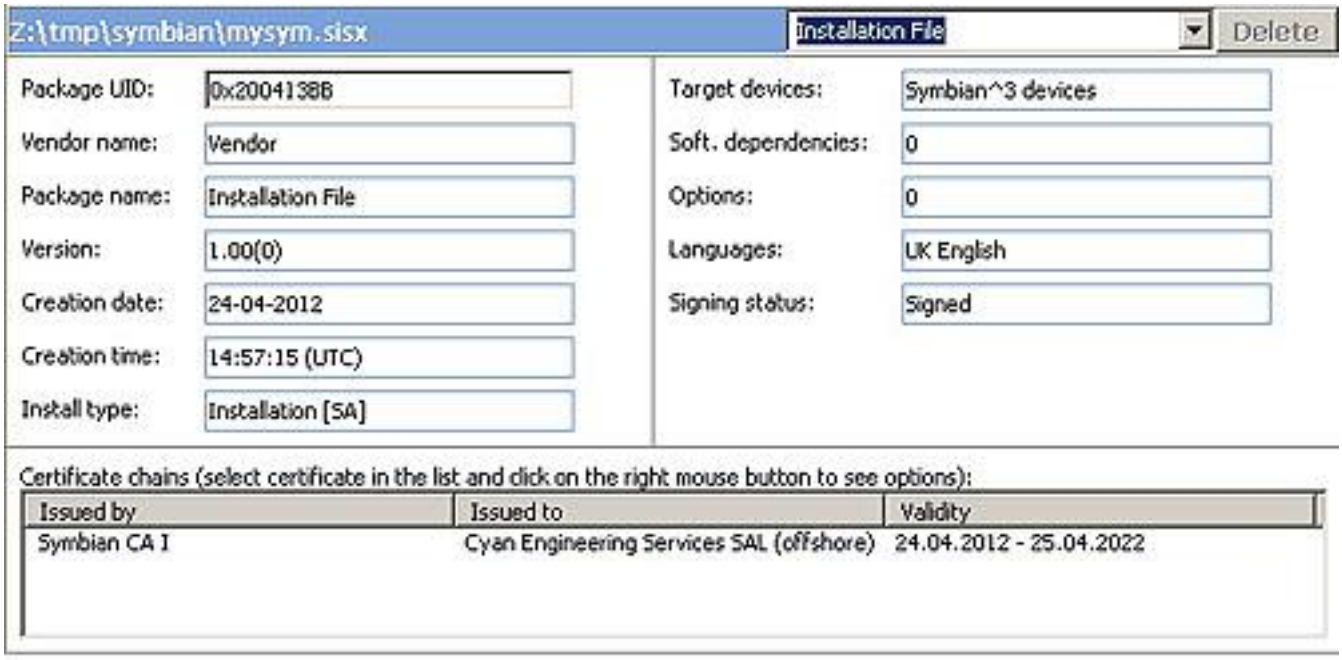
The file "c:\sys\bin\updater.exe" provides the main implant functionality. This requests the following capabilities[1]:

PowerMgmt
ReadDeviceData
TrustedUI
SwEvent
NetworkServices
ReadUserData
WriteUserData
Location

Of special note is the use of TrustedUI. As mentioned in the security section of the Nokia developer notes for Symbian:

> "Trusted UI dialogs are rare. They must be used only when confidentiality and security are critical: for instance for password dialogs. Normal access to the user interface and the screen does not require this."

The second sample ("mysym.sisx") identifies itself as "Installation File" and appears to be signed by the "Symbian CA I" for "Cyan Engineering Services SAL (offshore)," unlike the previous sample, which was registered to jd@cyanengineeringservices.com.



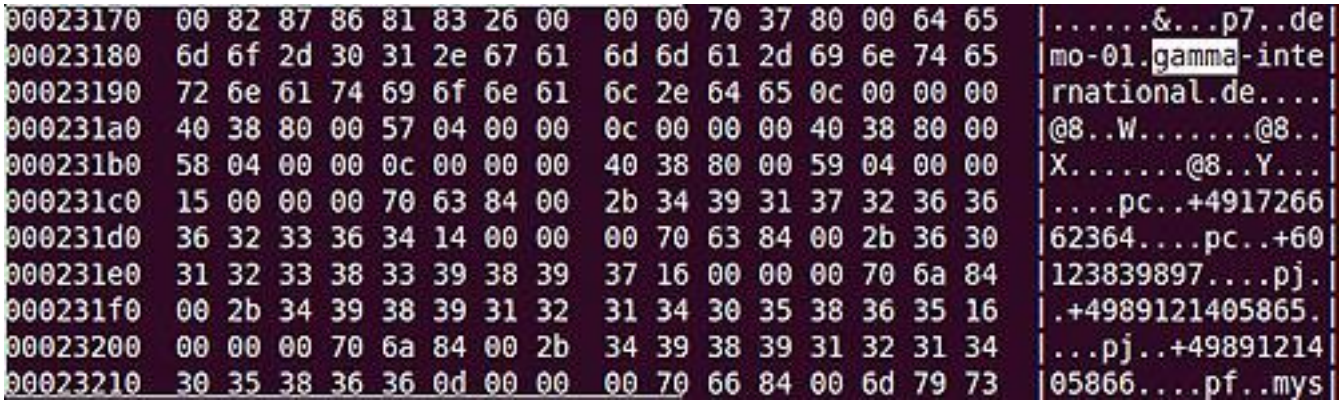
We identified "Cyan Engineering Services SAL (offshore)" as also listed as the registrant on the parked domain www.it-intrusion.com, (Created: 08-Dec-11, also with GoDaddy). However, **it-intrusion.com** does not have a protected registrant. The registrant is listed[2] as a company based in Beirut, Lebanon:

Cyan Engineering Services SAL (offshore)
Broadway Center, 7th Floor
Hamra Street - Chouran 1102-2050
Beirut, Beirut 00000
Lebanon
Domain Domain Name: IT-INTRUSION.COM
Created: 08-Dec-11
Expires: 08-Dec-13

Updated: 08-Dec-11
Administrative Contact: Debs, Johnny

The registrant information for Cyan Engineering Services SAL also connects to Gamma: the name "Johnny Debs" is associated with Gamma International: a Johnny Debs was listed as representing Gamma at the October 2011 Milpol in Paris, and the name occurs elsewhere in discussions of FinFisher.

Examination of this sample reveals the domain demo-01.gamma-international.de potentially indicating a demo or pre-customisation copy.



The phone number +60123839897 also shows up in the sample. It has a Malaysian country code.

Blackberry

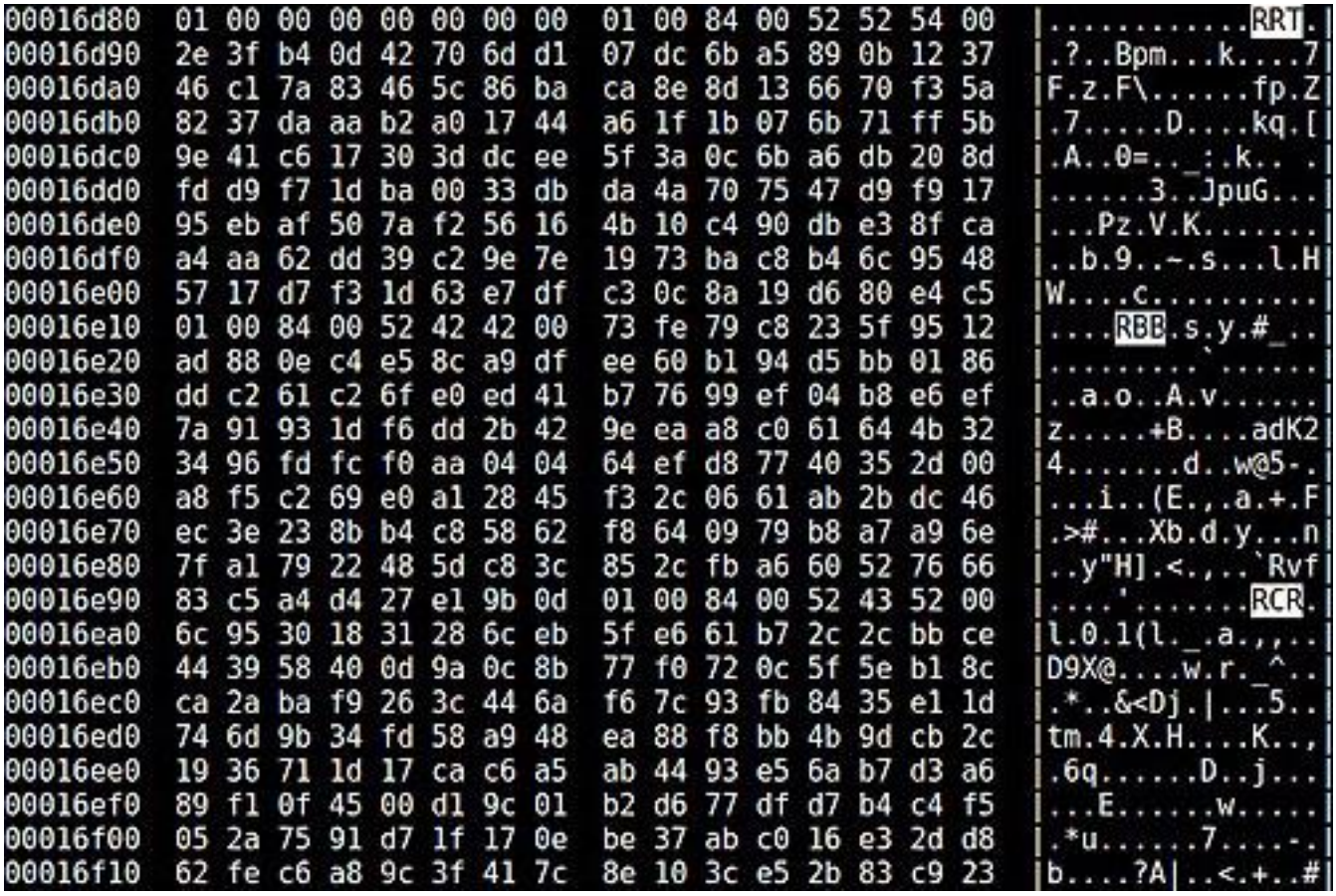
rlc_channel_mode_updater.cod
rlc_channel_mode_updater-1.cod
rlc_channel_mode_updater.jad

The identified samples contained the following files:

The .cod files are signed by RIM's RBB, RCR, and RRT keys. RBB stands for "RIM BlackBerry Apps API," which allows manipulation of BlackBerry apps, RCR stands for "RIM Crypto API," which allows access to crypto libraries, and RRT stands for "RIM Runtime API," which allows access to other phone functionality such as sending SMS messages.

The signature process is described in RIM's documentation [pdf] about the Blackberry Signing Authority. First, a developer registers a public key with the Blackberry Signing Authority. In order to obtain a signed application, the developer submits a signature request (including his identity and a hash of the binary) signed with his private key to the Signing Authority. The Signing Authority verifies that the signer is authorized to

make requests, and, if so, replies with a copy of the hash signed with the relevant RIM private key. The developer then appends the signature to his binary.



The .jad file contains the following hashes for the .cod files:

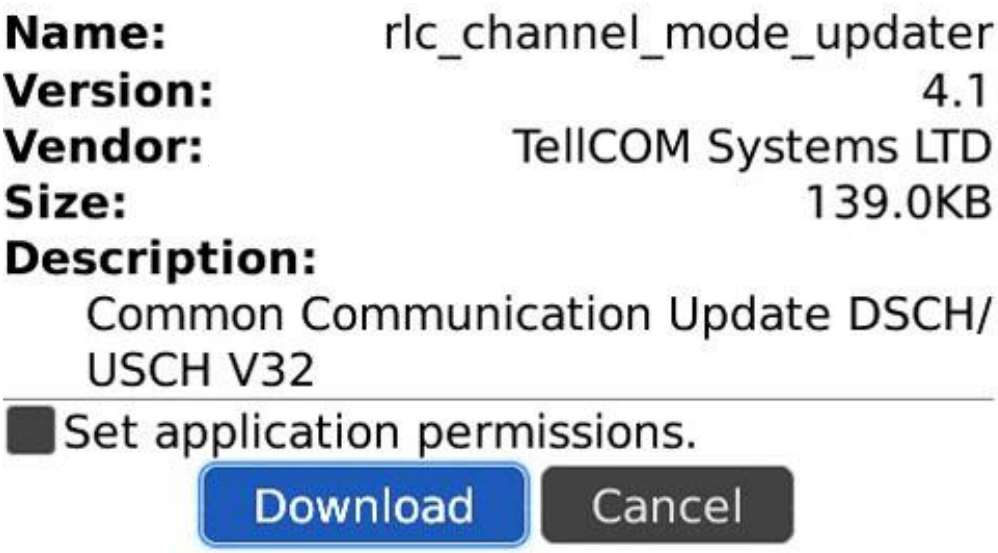
RIM-COD-SHA1-1: 2d 0a a2 b3 54 97 f7 35 fb 40 77 8e e1 ca 7f 8f 3e a0 aa 04
RIM-COD-SHA1: 0f 3b d8 d1 84 da 35 4e 10 94 89 c0 d6 08 70 ad 5e 7a f3 e0

The .jad file also contains a blob of base64 encoded data with the key "RIM-COD-Config." This data contains the URL of the command & control server, TCP ports, phone numbers to exfiltrate data to via SMS, identifiers for the Trojan and target, active modules, and various other configuration parameters.

Decoding this reveals the following servers and phone numbers:

118.xx.xx.186 - **Indonesia**
+6281310xxxxx4 - **Indonesia**
+49456xxxxx6 - **Germany**

Upon installation, the user is presented with the following screen:

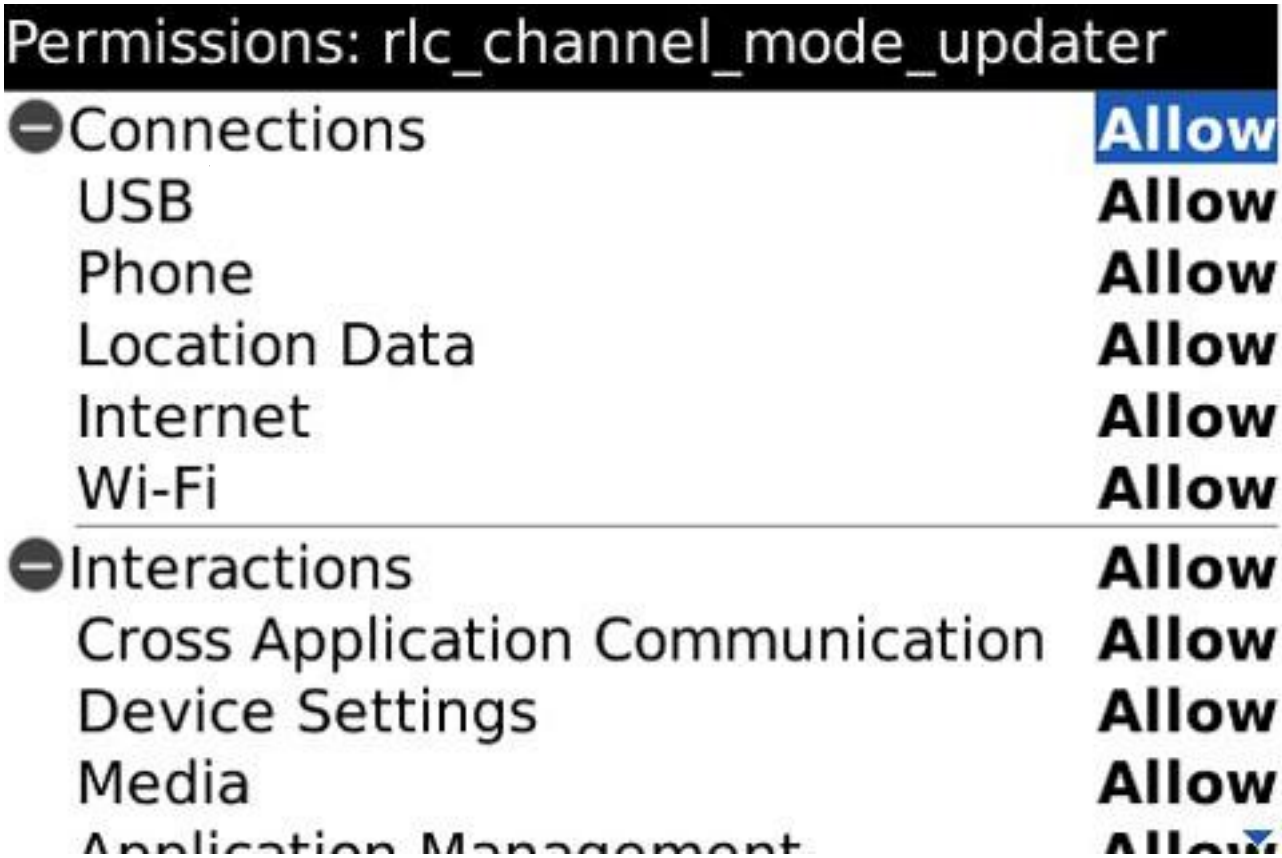


As evidenced by the above screenshot, the app is listed as:

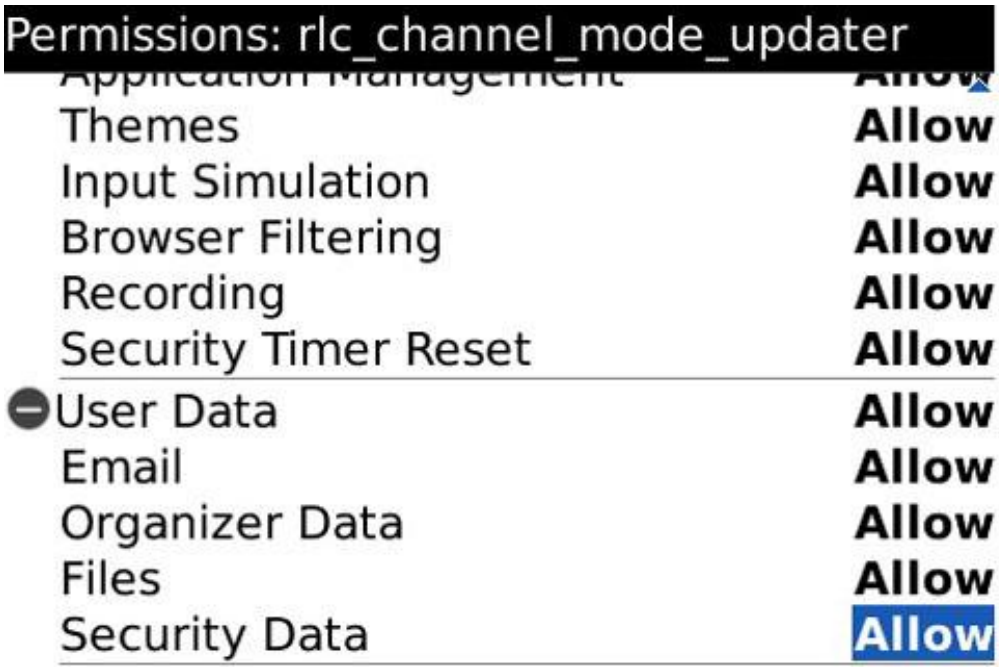
Directly after installing, the application requests enhanced permissions:

TellCOM Systems LTD Common Communication Update DSCH/USCH V32

The following screen pops up showing the requested permissions:



Scrolling down reveals:



After the user accepts these permissions, the sample attempts to connect to both Internet-based and SMS-based command & control servers. Another sample we analyzed appeared to write a debug log to the device's filesystem. The following information was observed written to the log regarding communication with command & control services.

> net.rmi.device.api.fsmbb.phone.PhoneInterface - connecting to http://demo-01.gamma-international.de:1111/ping/XXXXXXXXXXXX;deviceside=true failed:
> net.rim.device.cldc.io.dns.DNSException: DNS error DNS error

> net.rmi.device.api.fsmbb.core.com.protocol.HeartbeatProtocolSMS - Heartbeat type 11 (1346097705922)+ core hb content: XXXXX/123456783648138/666666553648138/12e/666/0/0///

> net.rmi.device.api.fsmbb.core.com.SMSCommunication - 1346097743 Success: texting to: //+XXXXXXXXXX msg: XXXXX

> net.rmi.device.api.fsmbb.core.com.protocol.HeartbeatProtocolSMS - Heartbeat type 11 (1346097705922)+ extended hb content: XXXXX/123456783648138/XXXXX/999/420/B9700 5.0.

> net.rmi.device.api.fsmbb.core.com.SMSCommunication - 1346097743 Success: texting to: //+XXXXXXXXXX msg: XXXXX

We decompiled the Blackberry sample. We provide a high-level overview of the more interesting classes that we successfully decompiled:

net.rmi.device.api.fsmbb.config.ApnDatabase
net.rmi.device.api.fsmbb.config.ApnDatabase$APN

These appeared to contain a database comprising the following GSM APNs. The significance of this database is that it only includes a small set of countries and providers:

Germany: web.vodafone.de, internet.t-mobile
Indonesia: indosatgprs, AXIS, telkomsel, www.xlgprs.net, 3gprs

Brazil: claro.com.br, wapgprs.oi.com.br, tim.br
Mexico: wap.telcel.com

net.rmi.device.api.fsmbb.core.AppMain

This appears to do the main app installation, as well as uninstallation. Installation includes negotiating for enhanced permissions, base64-decoding the "RIM-COD-Config" configuration, and setting up and installing the Configuration. If the configuration contains a "removal date," then automatic removal is scheduled for this time. Installation also involves instantiating "listener" modules, as specified below:

net.rmi.device.api.fsmbb.core.listener.AddressBookObserver

This appears to listen for changes to the address book. It implements the net.rim.blackberry.api.pim.PIMListListener interface.

net.rmi.device.api.fsmbb.core.listener.CallObserver.*

This implements:

net.rim.blackberry.api.phone.PhoneListener
net.rim.blackberry.api.phone.phonelogs.PhoneLogListener
net.rim.device.api.system.KeyListener

This module logs and manipulates phone events, and appears to enable "remote listening" functionality, where the FinSpy Master can silently call an infected phone to listen to conversation in its vicinity (this is referred to as a SpyCall in the code). The module has a facility to hide incoming calls by manipulating the UI, cancelling buzzer and vibration alerts, and toggling the backlight. Upon instantiation, the module calls "*43#" to enable call waiting. If a remote listening call from the master is active, then legitimate incoming calls will trigger call waiting. The module detects these legitimate incoming calls, and places the SpyCall call on call waiting, presenting the legitimate incoming call to the user.

net.rmi.device.api.fsmbb.core.listener.EmailObserver

This appears to record sent and received email messages.

net.rmi.device.api.fsmbb.core.listener.MessengerObserver (Module #68)

This seems to record BBM messages. It appears to do this by periodically checking the path "file:///store/home/user/im/BlackBerry Messenger/"

net.rmi.device.api.fsmbb.core.listener.SMSObserver

This module implements:

net.rim.blackberry.api.sms.SendListener
net.rim.blackberry.api.sms.OutboundMessageListener

Contrary to its name, OutboundMessageListener allows listening for both incoming and outgoing SMS messages. This module also checks for incoming SMS commands from the FinSpy Master. These commands can include an "emergency configuration" update, that can include new addresses and phone numbers for the FinSpy Master.

net.rmi.device.api.fsmbb.core.listener.WAObserver (Module #82)

This appears to monitor WhatsApp, the popular proprietary cross-platform messaging application. It locates the WhatsApp process ID by searching for module names that contain the string "WhatsApp."

At some point, the module calls getForegroundProcessId to see if the WhatsApp process ID is in the foreground. If so, it seems to take a screenshot of the WhatsApp application, via Display.Screenshot. It appears that this screenshot is checked via ".equals" to see if there is any new information on the WhatsApp screen. If there is new information, the screenshot is then JPEG encoded via JPEGEncodedImage.encode.

net.rmi.device.api.fsmbb.core.com.*

Appears to contain the mechanics of communication with the command & control server, including the plaintext TLV-based wire protocol.

Windows Mobile

The Windows Mobile samples we identified are:

2ccbfed8f05e6b50bc739c86ce4789030c6bc9e09c88b7c9d41cbcbde52a2455
507e6397e1f500497541b6958c483f8e8b88190407b307e997a4decd5eb0cd3a
1ff1867c1a55cf6247f1fb7f83277172c443442d174f0610a2dc062c3a873778

All the samples appeared similar, most likely belonging to the same branch release. The relevant parts of the binary are stored in five different resources:

- The first resource contains an OMA Client Provisioning XML file, which is used to store root certificates for running privileged/unprivileged code on the device. In this case it only contained some default example values shipped with Microsoft Windows Mobile SDK.
- The second resource contains the actual dropped payload which contains all the Trojan functionalities.
- The third resource contains a binary configuration file.
- The fourth and fifth resources contain two additional DLL files which are dropped along with the payload.

The main implant is dropped as "services.exe" with the libraries dropped as mapiwinarm.dll and mswservice.dll.

The payload has the following attributes:

File size: 186640 bytes
SHA256: 4b99053bc7965262e8238de125397d95eb7aac5137696c7044c2f07b175b5e7c

This is a multi-threaded and modular engine which is able to run and coordinate a series of events providing interception and monitoring capabilities. When the application starts, a core initialization function is invoked, responsible for preparing execution and launching the main thread.

The main thread consequently runs a set of core components on multiple threads:

- Routines responsible for handling the "heartbeat" notifications.
- Routines which control the execution of the Trojan and its components while monitoring the status of the device.
- A routine which can be used to "wake up" the device.
- A component which handles emergency SMS communications.
- A routine that initializes the use of the Radio Interface Layer.
- A core component that manages a set of surveillance modules.

The Trojan utilises a "Heartbeat Manager", which is a set of functions and routines that, depending on the status of the device or monitored events, communicates notifications back to the command and control server.

These beacons are sent according the following events:

- First beacon.
- A specified time interval elapsing.
- The device has low memory.
- The device has low battery.
- The device changed physical location.
- The Trojan has recorded data available.
- The device has connected to a cellular network.
- The device has a data link available.
- The device connects to a WiFi network.
- An incoming / outgoing call starts.
- The Mobile Country Code (MCC) or Mobile Network Code (MNC) ID changed.
- The Trojan is being uninstalled.
- The SIM changes.

Notifications are sent via SMS, 3G and WiFi, according to availability. Consistent with other platforms, the windows mobile version appears to use base64 encoding for all communications.

In response to such notifications, the implant is able to receive and process commands such as:

```
STOP_TRACKING_CMD
START_TRACKING_CMD
RESEND_FIRST_HEARTBEAT_TCPIP_CMD
RESEND_FIRST_HEARTBEAT_SMS_CMD
REMOVE_LICENSE_INFO_CMD
KEEP_CONNECTION_ALIVE_CMD IGNORED b/c it's an SMS answer
KEEP_CONNECTION_ALIVE_CMD
REMOVE_AT_AGENT_REQUEST_CMD
REMOVE_AT_MASTER_REQUEST_CMD
REMOVE_MAX_INFECTION_REACHED_CMD
```

The command and control server is defined in the configuration file found in the third resource of the dropper. In this sample, the sample connected to the domain: **demo-04.gamma-international.de**

This suggests that such sample is either a demo version or “unpackaged” version ready to be customized.

Together with a DNS or IP command and control server, each sample appears to be provided with two phone numbers which are used for SMS notifications.

The core surveillance and offensive capabilities of the Trojan are implemented through the use of several different modules. These modules are initialized by a routine we called ModulesManager, which loads and launches them in separate threads:

```
LDR     R3, =aTryToLoadModul ; "try to load module: %02X"
MOV     R1, #0
LDR     R2, =aModuleManageme ; "module-management:FxLoadModule"
MOV     R0, R6
STR     R4, [SP,#0x28+var_28]
BL      FinSpy_Log
ADD     R7, R6, R4,LSL#2
LDR     R3, [R7,#0x11C]
CMP     R3, #0
MOVNE   R3, #0
STRNE   R3, [R11,#var_24]
BNE     loc_20FE4
CMP     R4, #0x40
BEQ     FinSpy_MM_StartSpyCall
CMP     R4, #0x41
BEQ     FinSpy_MM_StartCallIntercept
CMP     R4, #0x42
BEQ     FinSpy_MM_StartSMS
CMP     R4, #0x43
BEQ     FinSpy_MM_StartLoader
CMP     R4, #0x45
BEQ     FinSpy_MM_StartTracking
CMP     R4, #0x46
BEQ     FinSpy_MM_StartCallLogs
CMP     R4, #0x60
BEQ     loc_20F30
LDR     R3, =aModule02xDoesn ; "module '%02X' doesn't exist"
LDR     R2, =aModuleManageme ; "module-management:FxLoadModule"
MOV     R1, #1
MOV     R0, R6
STR     R4, [SP,#0x28+var_28]
BL      FinSpy_Log
```

There are multiple modules available, including:

- AddressBook: Providing exfiltration of details from contacts stored in the local address book.
- CallInterception: Used to intercept voice calls, record them and store them for later transmission.

- PhoneCallLog: Exfiltrates information on all performed, received and missed calls stored in a local log file.
- SMS: Records all incoming and outgoing SMS messages and stores them for later transmission.
- Tracking: Tracks the GPS locations of the device.

Call Interception

In order to manipulate phone calls, the Trojan makes use of the functions provided by RIL.dll, the Radio Interface Layer.

Some of the functions imported and used can be observed below:

```
LDR     R1, =aRil_getcallwai ; "RIL_GetCallWaitingSettings"
MOV     R3, R0
LDR     R0, [R7,#0x14]  ; hModule
STR     R3, [R7,#0x6C]
BL      GetProcAddressW
LDR     R1, =aRil_setcallwai ; "RIL_SetCallWaitingStatus"
MOV     R3, R0
LDR     R0, [R7,#0x14]  ; hModule
STR     R3, [R7,#0x10C]
BL      GetProcAddressW
LDR     R1, =aRil_answer ; "RIL_Answer"
MOV     R3, R0
LDR     R0, [R7,#0x14]  ; hModule
STR     R3, [R7,#0xAC]
BL      GetProcAddressW
LDR     R1, =aRil_managecall ; "RIL_ManageCalls"
MOV     R3, R0
LDR     R0, [R7,#0x14]  ; hModule
STR     R3, [R7,#0x118]
BL      GetProcAddressW
LDR     R1, =aRil_getcalllis ; "RIL_GetCallList"
MOV     R3, R0
LDR     R0, [R7,#0x14]  ; hModule
STR     R3, [R7,#0xE0]
BL      GetProcAddressW
```

PhoneCallLog

In order to exfiltrate call logs, the Trojan uses functions provided by the Windows Mobile Phone Library.

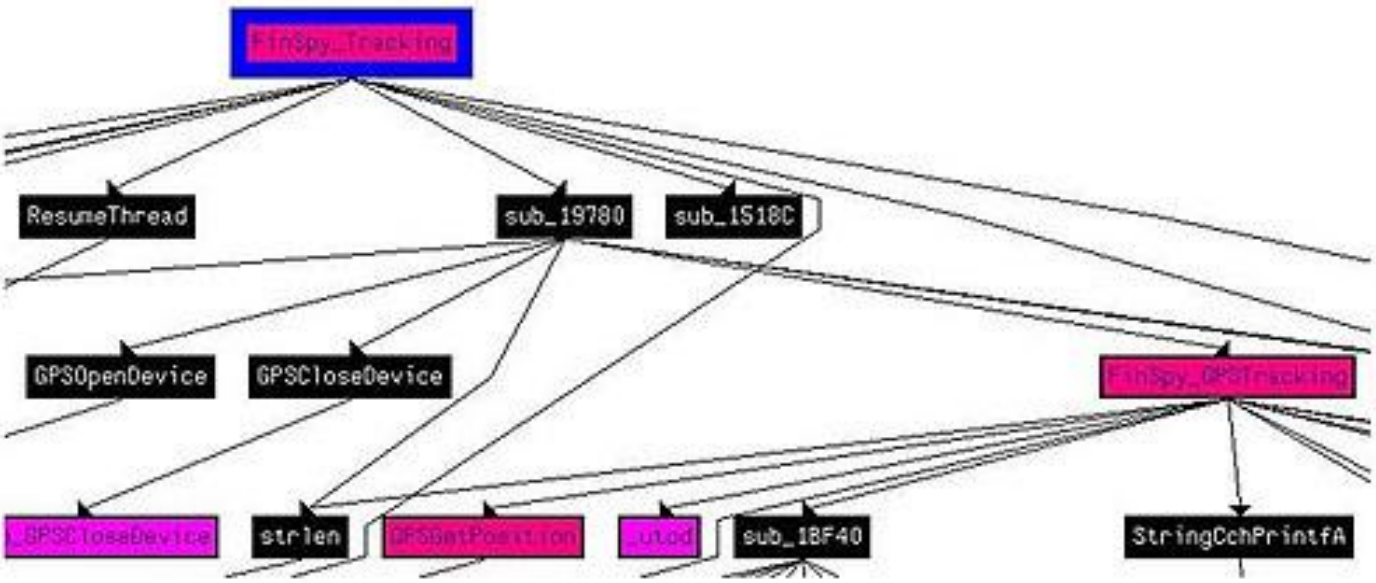
Using PhoneOpenCallLog() and PhoneGetCallLogEntry(), the implant is able to retrieve the following struct for each call being registered by the system:

```
typedef struct {
DWORD cbSize;
FILETIME ftStartTime;
FILETIME ftEndTime;
IOM iom;
BOOL fOutgoing:1;
BOOL fConnected:1;
BOOL fEnded:1;
BOOL fRoam:1;
CALLERIDTYPE cidt;
PTSTR pszNumber;
PTSTR pszName;
PTSTR pszNameType;
PTSTR pszNote;
DWORD dwLogFlags;
CEIOD iodContact;
CEPROPID pidProp;
} CALLLOGENTRY, * PCALLLOGENTRY;
```

This contains timestamps, numbers, names and other data associated with a call.

Tracking

The physical tracking of the device uses the GPS Intermediate Driver functions available on the Windows Mobile/CE platform:



After a successful GPSOpenDevice() call, it invokes GPSGetPosition() which gives access to a GPS_POSITION struct containing the following information:

```
typedef struct _GPS_POSITION {
DWORD dwVersion;
DWORD dwSize;
DWORD dwValidFields;
DWORD dwFlags;
SYSTEMTIME stUTCTime;
double dblLatitude;
double dblLongitude;
float flSpeed;
float flHeading;
double dblMagneticVariation;
float flAltitudeWRTSeaLevel;
float flAltitudeWRTEllipsoid;
GPS_FIX_QUALITY FixQuality;
GPS_FIX_TYPE FixType;
GPS_FIX_SELECTION SelectionType;
float flPositionDilutionOfPrecision;
float flHorizontalDilutionOfPrecision;
float flVerticalDilutionOfPrecision;
DWORD dwSatelliteCount;
DWORD rgdwSatellitesUsedPRNs[GPS_MAX_SATELLITES];
DWORD dwSatellitesInView;
DWORD rgdwSatellitesInViewPRNs[GPS_MAX_SATELLITES];
DWORD rgdwSatellitesInViewElevation[GPS_MAX_SATELLITES];
DWORD rgdwSatellitesInViewAzimuth[GPS_MAX_SATELLITES];
DWORD rgdwSatellitesInViewSignalToNoiseRatio[GPS_MAX_SATELLITES];
} GPS_POSITION, *PGPS_POSITION;
```

This provides the latitude and longitude of the current location of the device.

COMMAND AND CONTROL SERVER SCANNING RESULTS

Following up on our earlier analysis, we scanned IP addresses in several countries looking for FinSpy command & control servers. At a high level, our scans probed IP addresses in each country, and attempted to perform the handshake distinctive to the FinSpy command and control protocol. If a server responded to the handshake, we marked it as a FinSpy node. We expect to release our scanning tools with a more complete description of methodology in a follow-up blog post.

Our scanning yielded two key findings. First, we have identified several more countries where FinSpy Command and Control servers were operating. Scanning has thus far revealed two servers in **Brunei**, one in **Turkmenistan**'s Ministry of Communications, two in **Singapore**, one in the **Netherlands**, a new server in **Indonesia**, and a new server in **Bahrain**.

Second, we have been able to partially replicate the conclusions of an analysis by Rapid7, which reported finding FinSpy command & control servers in ten countries: Indonesia, Australia, Qatar, Ethiopia, Czech Republic, Estonia, USA, Mongolia, Latvia, and the UAE. We were able to confirm the presence of FinSpy on all of the servers reported by Rapid7 that were still available to be scanned. We confirmed FinSpy servers in **Indonesia**, **Ethiopia**, **USA**, **Mongolia**, and the UAE. The remaining servers were down at scanning time. We also noted that the server in the USA appeared to be an IP-layer proxy (e.g., in the style of Network Address Translation)[3].

Rapid7's work exploited a temporary anomaly in FinSpy command & control servers. Researchers at Rapid7 noticed that the command & control server in Bahrain responded to HTTP requests with the string "Hallo Steffi." This behavior did not seem to be active on Bahrain's server prior to the release of our analysis. Rapid7 looked at historical scanning information, and noticed that servers in ten other countries had responded to HTTP requests with "Hallo Steffi" at various times over the previous month. While the meaning of this string and the reason for the temporary anomaly are unknown, a possible explanation is that this was a testing deployment of a server update, and the "Hallo Steffi" message indicated successful receipt of the update. After the publication of Rapid7's analysis, the behavior began to disappear from FinSpy servers.

DETAILS OF OBSERVED SERVERS

Table 1: New Servers

Country	IP	Ports	Owner
Singapore	203.175.168.2	21, 53, 443, 4111	HostSG
Singapore	203.211.137.105	21, 53, 80, 443, 4111	M1 CONNECT PTE. LTD.
Bahrain	89.148.15.15	22, 53, 80, 443, 4111	Batelco
Turkmenistan	217.174.229.82	22, 53, 80, 443, 4111, 9111	Ministry of Communications
Brunei	119.160.172.187	21	Telekom Brunei
Brunei	119.160.128.219	4111, 9111	Telekom Brunei
Indonesia	112.78.143.34	22, 53, 80, 443, 9111	Biznet ISP
Netherlands	164.138.28.2	80, 1111	Tilaa VPS Hosting

Table 2: Confirmed Rapid7 Servers

Country	IP	Ports	Owner
USA	54.248.2.220	80	Amazon EC2
Indonesia	112.78.143.26	22, 25, 53, 80, 443, 4111	Biznet ISP
Ethiopia	213.55.99.74	22, 53, 80, 443, 4111, 9111	Ethio Telecom
Mongolia	202.179.31.227	53, 80, 443	Mongolia Telecom
UAE	86.97.255.50	21, 22, 53, 443, 4111	Emirates Telecommunications Corporation

It is interesting to note that the USA server on EC2 appeared to be an IP-layer proxy. This judgment was made on the basis of response time comparisons[4].

CONCLUSIONS & RECOMMENDATIONS

The analysis we have provided here is a continuation of our efforts to analyze what appear to be parts of the FinFisher product portfolio. We found evidence of the functionality that was specified in the FinFisher promotional materials. The tools and company names (e.g. Cyan Engineering Services SAL) found in their certificates also suggest interesting avenues for future research.

These tools provide substantial surveillance functionality; however, we'd like to highlight that, without exploitation of the underlying platforms, all of the samples we've described require some form of interaction to install. As with the previously analyzed FinSpy tool this might involve some form of socially engineered e-mail or other delivery, prompting unsuspecting users to execute the program. Or, it might involve covert or coercive physical installation of the tool, or use of a user's credentials to perform a third-party installation.

We recommend that all users run Anti-Virus software, promptly apply (legitimate) updates when they become available, use screen locks, passwords and device encryption (when available). Do not run untrusted applications and do not allow third parties access to mobile devices.

As part of our ongoing research, we have notified vendors, as well as members of the AV community.

FOOTNOTES

[1] A list of Nokia capabilities can be found here.
[2] http://www.whoisentry.com/domain/it-intrusion.com
[3] See Appendix A.
[4] See Appendix A.

APPENDIX A

The server was serving FinSpy on port 80, and SSH on port 22. We measured the SYN/ACK RTT on both ports and compared. The results for port 80:

```
hping -S -p 80 54.248.2.220
HPING 54.248.2.220 (wlan0 54.248.2.220): S set, 40 headers + 0 data bytes
len=44 ip=54.248.2.220 ttl=24 DF id=0 sport=80 flags=SA seq=0 win=5840 rtt=1510.2 ms
len=44 ip=54.248.2.220 ttl=23 DF id=0 sport=80 flags=SA seq=1 win=5840 rtt=740.4 ms
len=44 ip=54.248.2.220 ttl=25 DF id=0 sport=80 flags=SA seq=2 win=5840 rtt=753.4 ms
len=44 ip=54.248.2.220 ttl=24 DF id=0 sport=80 flags=SA seq=3 win=5840 rtt=1001.6 ms
```

The results for port 22:

```
hping -S -p 22 54.248.2.220
HPING 54.248.2.220 (wlan0 54.248.2.220): S set, 40 headers + 0 data bytes
len=44 ip=54.248.2.220 ttl=49 DF id=0 sport=22 flags=SA seq=0 win=5840 rtt=125.7 ms
len=44 ip=54.248.2.220 ttl=49 DF id=0 sport=22 flags=SA seq=1 win=5840 rtt=124.3 ms
len=44 ip=54.248.2.220 ttl=49 DF id=0 sport=22 flags=SA seq=2 win=5840 rtt=123.3 ms
len=44 ip=54.248.2.220 ttl=50 DF id=0 sport=22 flags=SA seq=3 win=5840 rtt=127.2 ms
```

The comparison reveals that port 80 TCP traffic was likely being proxied to a different computer.

ACKNOWLEDGEMENTS

This is a Morgan Marquis-Boire and Bill Marczak production.

Windows mobile sample analysis by Claudio Guarnieri.

Additional Analysis

Thanks to Pepi Zawodsky for OSX expertise and assistance.
Thanks to Jon Larimer and Sebastian Porst for Android expertise.

Additional Thanks

Special thanks to John Scott-Railton.
Additional thanks to Marcia Hofmann and the Electronic Frontier Foundation.
Tip of the hat to John Adams for scanning advice.

MEDIA COVERAGE

Bloomberg
The Next Web
Financial Post (The National Post)
Washington Post

About Morgan Marquis-Boire

Morgan Marquis-Boire is a Technical Advisor at the Citizen Lab, Munk School of Global Affairs, University of Toronto. He works as a Security Engineer at Google specializing in Incident Response, Forensics and Malware Analysis.

About Bill Marczak

Bill Marczak is a computer science Ph.D student at UC Berkeley. He is a founding member of Bahrain Watch, a monitoring and advocacy group that seeks to promote effective, accountable, and transparent governance in Bahrain through research and evidence-based activism.

About Claudio Guarnieri

Claudio Guarnieri is a Security Researcher at Rapid7. He's daily involved with general Internet badness and his specialties span from malware analysis to botnets tracking and cybercrime intelligence. He's a core member of The Honeynet Project and The Shadowserver Foundation, two no-profit organizations devoted to making Internet a safer place.