UNIVERSITY OF TORONTO

MUNK SCHOOL OF GLOBAL AFFAIRS

Join the Global Conversation

The Citizen Lab

Research Brief
July 2012

From Bahrain with Love:
FinFisher's Spy Kit Exposed?

Author: Morgan Marquis-Boire

INTRODUCTION

The FinFisher Suite is described by its distributors, Gamma International UK Ltd., as "Governmental IT Intrusion and Remote Monitoring Solutions." [1] The toolset first gained notoriety after it was revealed that the Egyptian Government's state security apparatus had been involved in negotiations with Gamma International UK Ltd. over the purchase of the software. Promotional materials have been leaked that describe the tools as providing a wide range of intrusion and monitoring capabilities.[2] Despite this, however, the toolset itself has not been publicly analyzed.

This post contains analysis of several pieces of malware obtained by Vernon Silver of Bloomberg News that were sent to Bahraini pro-democracy activists in April and May of this year. The purpose of this work is identification and classification of the malware to better understand the actors behind the attacks and the risk to victims. In order to accomplish this, we undertook several different approaches during the investigation.

As well as directly examining the samples through static and dynamic analysis, we infected a virtual machine (VM) with the malware. We monitored the filesystem, network, and running operating system of the infected VM.

This analysis suggests the use of "Finspy", part of the commercial intrusion kit, Finfisher, distributed by Gamma International.

DELIVERY

This section describes how the malware was delivered to potential victims using e-mails with malicious attachments.

In early May, we were alerted that Bahraini activists were targeted with apparently malicious e-mails. The emails ostensibly pertained to the ongoing turmoil in Bahrain, and encouraged recipients to open a series of suspicious attachments. The screenshot below is indicative of typical message content:

----- Forwarded Message -----
From: Melissa Chan <melissa.aljazeera@gmail.com>
To:
Sent: Tuesday, 8 May 2012, 8:52
Subject: Torture reports on Nabeel Rajab

Acting president Zainab Al Khawaja for Human Rights Bahrain reports of torture on Mr. Nabeel Rajab after his recent arrest.

Please check the attached detailed report along with torture images.

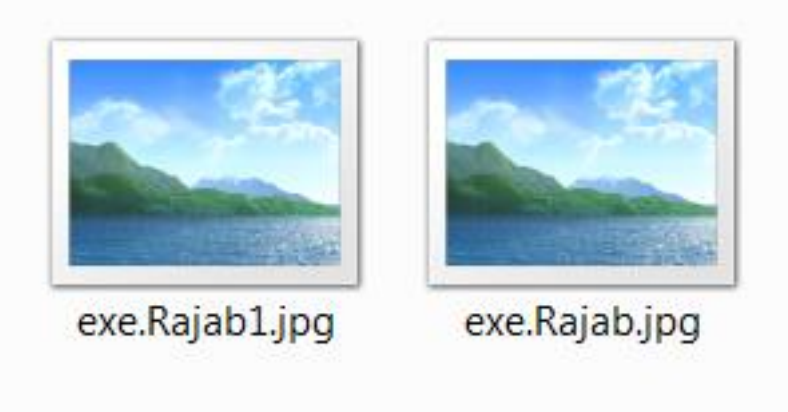
The attachments to the e-mails we have been able to analyze were typically .rar files, which we found to contain malware. Note that the apparent sender has an e-mail address that indicates that it was being sent by "Melissa Chan," who is a real correspondent for Aljazeera English. We suspect that the e-mail address is not her real address.[3] The following samples were examined:

324783fbc33ec117f971cca77ef7ceaf7ce229a74edd6e2b3bd0effd9ed10dcc rar.الفعاليات
c5b39d98c85b21f8ac1bedd91f0b6510ea255411cf19c726545c1d0a23035914 _gpj.ArrestedXSuspects.rar
c5b37bb3620d4e7635c261e5810d628fc50e4ab06b843d78105a12cfbbea40d7
KingXhamadXonXofficialXvisitXtoX.rar
80fb86e265d44fbabac942f7b26c973944d2ace8a8268c094c3527b83169b3cc MeetingXAgenda.rar
f846301e7f190ee3bb2d3821971cc2456617edc2060b07729415c45633a5a751 Rajab.rar

These contained executables masquerading as picture files or documents:

49000fc53412bfda157417e2335410cf69ac26b66b0818a3be7eff589669d040 dialoge.exe
cc3b65a0f559fa5e6bf4e60eef3bffe8d568a93dbb850f78bdd3560f38218b5c exe.Rajab1.jpg
39b325bd19e0fe6e3e0fca355c2afddfe19cdd14ebda7a5fc96491fc66e0faba exe.image1.jpg
e48bfeab2aca1741e6da62f8b8fc9e39078db574881691a464effe797222e632 exe.Rajab.jpg
2ec6814e4bad0cb03db6e241aabdc5e59661fb580bd870bdb50a39f1748b1d14 exe.Arrested Suspects.jpg
c29052dc6ee8257ec6c74618b6175abd6eb4400412c99ff34763ff6e20bab864 News about the existence of a new dialogue between AlWefaq & Govt..doc

The emails generally suggested that the attachments contained political content of interest to pro-democracy activists and dissidents. In order to disguise the nature of the attachments a malicious usage of the "righttoleftoverride" (RLO) character was employed. The RLO character (U+202e in unicode) controls the positioning of characters in text containing characters flowing from right to left, such as Arabic or Hebrew. The malware appears on a victim's desktop as "exe.Rajab1.jpg" (for example), along with the default Windows icon for a picture file without thumbnail. But, when the UTF-8 based filename is displayed in ANSI, the name is displayed as "gpj.1bajaR.exe". Believing that they are opening a harmless ".jpg", victims are instead tricked into running an executable ".exe" file.[4]



Upon execution these files install a multi-featured trojan on the victim's computer. This malware provides the attacker with clandestine remote access to the victim's machine as well as comprehensive data harvesting and exfiltration capabilities.

INSTALLATION

This section describes how the malware infects the target machine.

The malware displays a picture as expected. This differs from sample to sample. The sample "Arrested Suspects.jpg" ("gpj.stcepsuS detserrA.exe") displays:

It additionally creates a directory (which appears to vary from sample to sample):

```
C:\Documents and Settings\XPMUser\Local Settings\Temp\TMP51B7AFEF
```

It copies itself there (in this case the malware appears as “Arrested Suspects.jpg”) where it is renamed:

```
C:\Documents and Settings\XPMUser\Local Settings\Temp\TMP51B7AFEF\Arrested Suspects.jpg” =>
C:\Documents and Settings\XPMUser\Local Settings\Temp\TMP51B7AFEF\tmpD.tmp
```

Then it drops the following files:

```
C:\DOCUME~1\%USER%\LOCALS~1\Temp\delete.bat
C:\DOCUME~1\%USER%\LOCALS~1\Temp\driverw.sys
```

It creates the folder (the name of which varies from host to host):

```
C:\Documents and Settings\%USER%\Application Data\Microsoft\Installer\{5DA45CC9-D840-47CC-9F86-FD2E9A718A41}
```

This process is observable on the filesystem timeline of the infected host (click image to enlarge):

“driverw.sys” is loaded and then “delete.bat” is run which deletes the original payload and itself. It then infects existing operating system processes, connects to the command and control server, and begins data harvesting and exfiltration.

Examining the memory image of a machine infected with the malware shows that a technique for infecting processes known as "**process hollowing"** is used. For example, the memory segment below from the "winlogon.exe" process is marked as executable and writeable:

```
Process: winlogon.exe Pid: 424 Address: 0x1af0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 19, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x01af0000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x01af0010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x01af0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x01af0030  00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00 00   ................
```

Here the malware starts a new instance of a legitimate process such as "winlogon.exe" and before the process's first thread begins, the malware de-allocates the memory containing the legitimate code and injects malicious code in its place. Dumping and examining this memory segment reveals the following strings in the infected process:

```
00003960  47 4e 55 20 4d 50 3a 20  43 61 6e 6e 6f 74 20 61  |GNU MP: Cannot a|
00003970  6c 6c 6f 63 61 74 65 20  6d 65 6d 6f 72 79 20 28  |llocate memory (|
00003980  73 69 7a 65 3d 25 75 29  0a 00 00 00 47 4e 55 20  |size=%u)....GNU |
00003990  4d 50 3a 20 43 61 6e 6e  6f 74 20 72 65 61 6c 6c  |MP: Cannot reall|
000039a0  6f 63 61 74 65 20 6d 65  6d 6f 72 79 20 28 6f 6c  |ocate memory (ol|
000039b0  64 5f 73 69 7a 65 3d 25  75 20 6e 65 77 5f 73 69  |d_size=%u new_si|
000039c0  7a 65 3d 25 75 29 0a 00  79 3a 5c 6c 73 76 6e 5f  |ze=%u)..y:\lsvn_|
000039d0  62 72 61 6e 63 68 65 73  5c 66 69 6e 73 70 79 76  |branches\finspyv|
000039e0  34 2e 30 31 5c 66 69 6e  73 70 79 76 32 5c 73 72  |4.01\finspyv2\sr|
000039f0  63 5c 6c 69 62 73 5c 6c  69 62 67 6d 70 5c 6d 70  |c\libs\libgmp\mp|
00003a00  6e 2d 74 64 69 76 5f 71  72 2e 63 00 63 20 3d 3d  |n-tdiv_qr.c.c ==|
00003a10  20 30 00 00 00 00 00 00  01 02 03 03 04 04 04 04  | 0...............|
```

Note the string:

y:\lsvn_branches\finspyv4.01\finspyv2\src\libs\libgmp\mpn-tdiv_qr.c

This file seems to correspond to a file in the GNU Multi-Precision arithmetic library:
http://gmplib.org:8000/gmp/file/b5ca16212198/mpn/generic/tdiv_qr.c

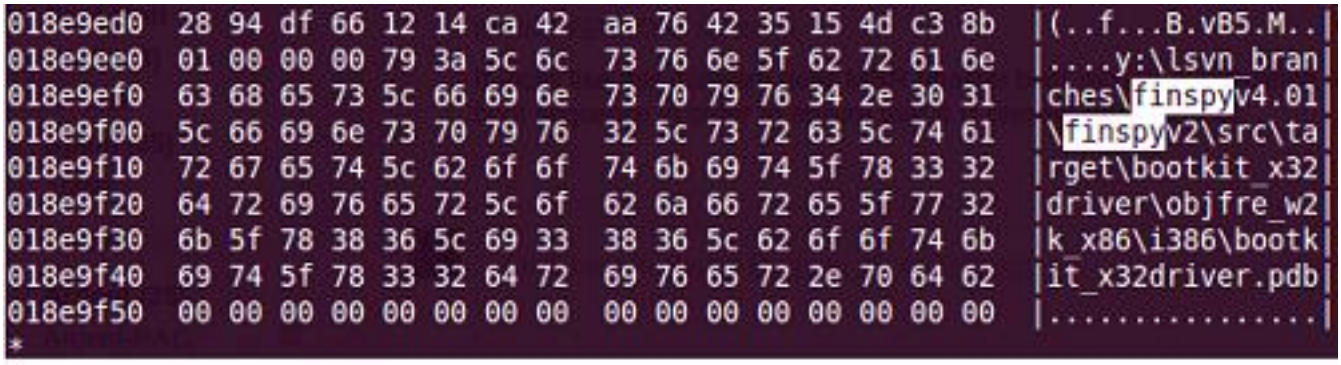
The process “svchost.exe” was also found to be infected in a similar manner:

```
Process: svchost.exe Pid: 760 Address: 0xbd0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00bd0000  8b ff 55 8b ec 68 40 47 f1 73 c3 8b ff 55 8b ec   ..U..h@G.s...U..
0x00bd0010  68 c0 68 f3 73 c3 8b ff 55 8b ec 68 ae 8e b4 76   h.h.s...U..h...v
0x00bd0020  c3 8b ff 55 8b ec 68 e2 c0 b5 76 c3 8b ff 55 8b   ...U..h...v...U.
0x00bd0030  ec 68 ff c2 b5 76 c3 8b ff 55 8b ec 68 3d c3 b5   .h...v...U..h=..

0xbd0000 8bff            MOV EDI, EDI
0xbd0002 55              PUSH EBP
0xbd0003 8bec            MOV EBP, ESP
0xbd0005 684047f173      PUSH DWORD 0x73f14740
0xbd000a c3              RET
0xbd000b 8bff            MOV EDI, EDI
0xbd000d 55              PUSH EBP
0xbd000e 8bec            MOV EBP, ESP
0xbd0010 68c068f373      PUSH DWORD 0x73f368c0
0xbd0015 c3              RET
0xbd0016 8bff            MOV EDI, EDI
0xbd0018 55              PUSH EBP
0xbd0019 8bec            MOV EBP, ESP
0xbd001b 68ae8eb476      PUSH DWORD 0x76b48eae
0xbd0020 c3              RET
0xbd0021 8bff            MOV EDI, EDI
0xbd0023 55              PUSH EBP
0xbd0024 8bec            MOV EBP, ESP
0xbd0026 68e2c0b576      PUSH DWORD 0x76b5c0e2
0xbd002b c3              RET
0xbd002c 8bff            MOV EDI, EDI
0xbd002e 55              PUSH EBP
0xbd002f 8bec            MOV EBP, ESP
0xbd0031 68ffc2b576      PUSH DWORD 0x76b5c2ff
0xbd0036 c3              RET
0xbd0037 8bff            MOV EDI, EDI
0xbd0039 55              PUSH EBP
0xbd003a 8bec            MOV EBP, ESP
0xbd003c 68              DB 0x68
0xbd003d 3d              DB 0x3d
0xbd003e c3              RET
0xbd003f b5              DB 0xb5
```

Further examination of the memory dump also reveals the following:



This path appears to reference the functionality that the malware uses to modify the boot sequence to enable persistence:

> y:\lsvn_branches\finspyv4.01\finspyv2\src\target\bootkit_x32driver\objfre_w2k_x86\i386\bootkit_x32driver.pdb

A pre-infection vs post-infection comparison of the infected VM shows that the Master Boot Record (MBR) was modified by code injected by the malware.

The strings found in memory "finspyv4.01" and "finspyv2" are particularly interesting. The FinSpy tool is part of the FinFisher intrusion and monitoring toolkit.[5]

OBFUSCATION AND EVASION

This section describes how the malware is designed to resist analysis and evade identification.

The malware employs a myriad of techniques designed to evade detection and frustrate analysis. While investigation into this area is far from complete, we discuss several discovered methods as examples of the lengths taken by the developers to avoid identification.

A virtualised packer is used. This type of obfuscation is used by those that have "strong motives to prevent their malware from being analyzed".[6]

This converts the native x86 instructions of the malware into another custom language chosen from one of 11 code templates. At run-time, this is interpreted by an obfuscated interpreter customized for that particular language. This virtualised packer was not recognised and appears to be bespoke.

Several anti-debugging techniques are used. This section of code crashes the popular debugger, OllyDbg.

```
.text:00401683 finit
.text:00401686 fld ds:tbyte_40168E
.text:0040168C jmp short locret_401698
----------------------------------------------------------------
.text:0040168E tbyte_40168E dt 9.22337203685477580 75e18
----------------------------------------------------------------
.text:00401698 locret_401698:
.text:00401698 retn
```

This float value causes OllyDbg to crash when trying to display its value. A more detailed explanation of this can be found here.

To defeat DbgBreakPoint based debuggers, the malware finds the address of DbgBreakPoint, makes the page EXECUTE_READWRITE and writes a NOP on the entry point of DbgBreakPoint.

The malware checks via PEB to detect whether or not it is being debugged, and if it is it returns a random address.

The malware calls ZwSetInformationThread with ThreadInformationClass set to 0x11, which causes the thread to be detached from the debugger.

The malware calls ZwQueryInformationProcess with ThreadInformationClass set to 0x(ProcessDebugPort) and 0x1e (ProcessDebugObjectHandle) to detect the presence of a debugger. If a debugger is detected it jumps to a random address. ZwQueryInformationProcess is also called to check the DEP status on the current process, and it disables it if it's found to be enabled.

The malware deploys a granular solution for Antivirus software, tailored to the AV present on the infected machine. The malware calls ZwQuerySystemInformation to get ProcessInformation and ModuleInformation. The malware then walks the list of processes and modules looking for installed AV software. Our analysis indicates that the malware appears to have different code to Open/Create process and inject for each AV solution. For some Anti-Virus software this even appears to be version dependent. The function "ZwQuerySystemInformation" is also hooked by the malware, a technique frequently used to allow process hiding:

```
********************************************************************************
Hook mode: Usermode
Hook type: Inline/Trampoline
Process: 628 (svchost.exe)
Victim module: ntdll.dll (0x7c900000 - 0x7c9b2000)
Function: ntdll.dll!ZwQuerySystemInformation at 0x7c90d92e
Hook address: 0xfd34b8
Hooking module: <unknown>

Disassembly(0):
0x7c90d92e e9855b6c84       JMP 0xfd34b8
0x7c90d933 ba0003fe7f       MOV EDX, 0x7ffe0300
0x7c90d938 ff12             CALL DWORD [EDX]
0x7c90d93a c21000           RET 0x10
0x7c90d93d 90               NOP
0x7c90d93e b8ae000000       MOV EAX, 0xae
0x7c90d943 ba               DB 0xba
0x7c90d944 0003             ADD [EBX], AL

Disassembly(1):
0xfd34b8 8bff             MOV EDI, EDI
0xfd34ba 55               PUSH EBP
0xfd34bb 8bec             MOV EBP, ESP
0xfd34bd 56               PUSH ESI
0xfd34be ff7514           PUSH DWORD [EBP+0x14]
0xfd34c1 8b750c           MOV ESI, [EBP+0xc]
0xfd34c4 ff7510           PUSH DWORD [EBP+0x10]
0xfd34c7 56               PUSH ESI
0xfd34c8 ff7508           PUSH DWORD [EBP+0x8]
0xfd34cb ff               DB 0xff
0xfd34cc 15               DB 0x15
0xfd34cd 9c               PUSHF
0xfd34ce 9d               POPF
0xfd34cf fd               STD
```

DATA HARVESTING AND ENCRYPTION

This section describes how the malware collects and encrypts data from the infected machine.

Our analysis showed that the malware collects a wide range of data from an infected victim. The data is stored locally in a hidden directory, and is disguised with encryption prior to exfiltration.

"C:\Windows\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}."

On the reference victim host, the directory was:

We conducted forensic examination of the files created in this directory and identified a wide range of data collected. Files in this directory were found to be screenshots, keylogger data, audio from Skype calls, passwords and more. For the sake of brevity we include a limited set of examples here.

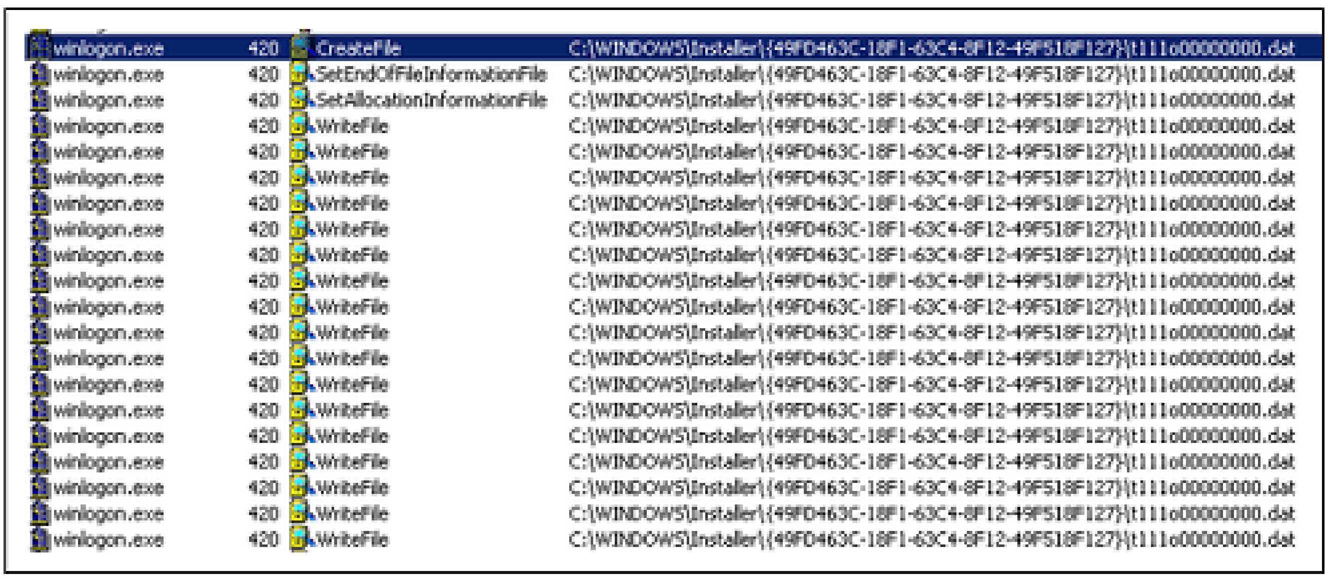
The malware attempts to locate the configuration and password store files for a variety browsers and chat clients as seen below:

rundll32.exe	3996	QueryOpen	C:\Documents and Settings\XPMUser\Application Data	SUCCESS
rundll32.exe	3996	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Profiles	NAME NOT FOUND
rundll32.exe	3996	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Thunderbird\Profiles	PATH NOT FOUND
rundll32.exe	3996	QueryOpen	C:\Documents and Settings\XPMUser\Local Settings\Application Data	SUCCESS
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data	SUCCESS
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Trillian\users\global	PATH NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Profiles	NAME NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\.gaim	NAME NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\.purple	NAME NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Miranda	NAME NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Local Settings\Application Data	SUCCESS
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\MySpace\IM\users.txt	PATH NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Digsby\digsby.dat	PATH NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\history.dat	NAME NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\places.sqlite	SUCCESS
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\nssckbi.dll	NAME NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\nssckbi.dll	NAME NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\signons.txt	NAME NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\signons2.txt	NAME NOT FOUND
rundll32.exe	4024	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\signons3.txt	NAME NOT FOUND
rundll32.exe	4060	QueryOpen	C:\Documents and Settings\XPMUser\Application Data	SUCCESS
rundll32.exe	4060	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\history.dat	NAME NOT FOUND
rundll32.exe	4060	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\places.sqlite	SUCCESS
rundll32.exe	4060	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\nssckbi.dll	NAME NOT FOUND
rundll32.exe	4060	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\nssckbi.dll	NAME NOT FOUND
rundll32.exe	4060	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\signons.sqlite	SUCCESS
rundll32.exe	4060	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\signons.sqlite…	NAME NOT FOUND
rundll32.exe	4060	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\signons.sqlite…	NAME NOT FOUND
rundll32.exe	4060	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\signons.sqlite…	NAME NOT FOUND
rundll32.exe	4060	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Mozilla\Firefox\Profiles\yz9d0pnf.default\signons.sqlite…	NAME NOT FOUND
rundll32.exe	4068	QueryOpen	C:\Documents and Settings\XPMUser\Local Settings\Application Data	SUCCESS
rundll32.exe	4068	QueryOpen	C:\Documents and Settings\XPMUser\Local Settings\Application Data\Google\Chrome\User Data\Default\Web …	PATH NOT FOUND
rundll32.exe	4068	QueryOpen	C:\Documents and Settings\XPMUser\Local Settings\Application Data\Google\Chrome\User Data\Default\Login…	PATH NOT FOUND
rundll32.exe	4080	QueryOpen	C:\Documents and Settings\XPMUser\Application Data	SUCCESS
rundll32.exe	4080	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Opera\Opera\wand.dat	PATH NOT FOUND
rundll32.exe	4080	QueryOpen	C:\Documents and Settings\XPMUser\Application Data\Opera\Opera7\profile\wand.dat	PATH NOT FOUND
rundll32.exe	4088	QueryOpen	C:\Documents and Settings\XPMUser\Local Settings\Application Data	SUCCESS

We observed the creation of the file “t111o00000000.dat” in the data harvesting directory, as shown in the filesystem timeline below:

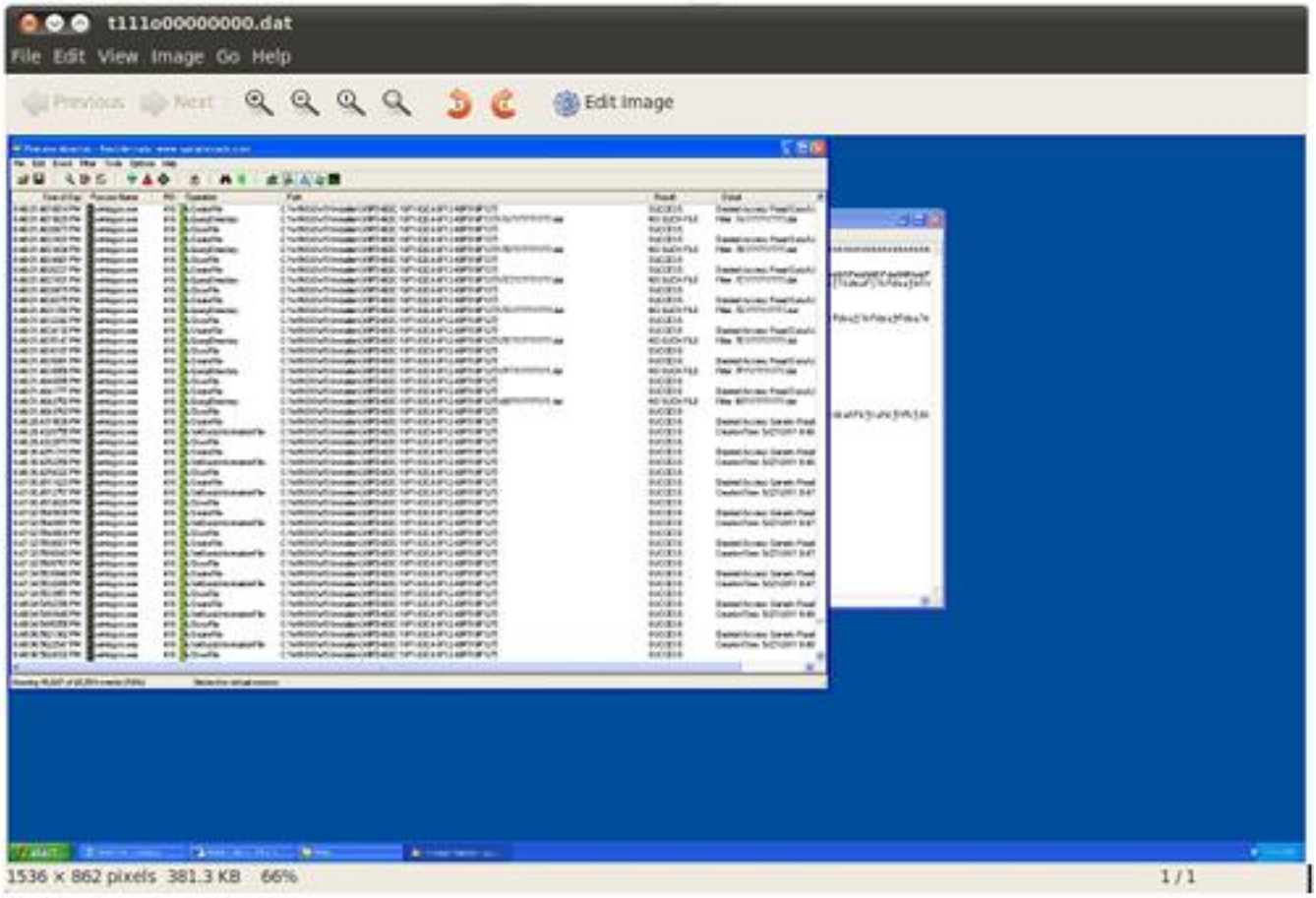
Thu Jun 14 2012 12:31:34 52719 mac. r/rr-xr-xr-x 0 0 26395-128-5 C:/WINDOWS/Installer/{49FD463C-18F1-63C4-8F12-49F518F127}/09e493e2-05f9-4899-b661-c52f3554c644
Thu Jun 14 2012 12:32:18 285691 ...b r/rrwxrwxrwx 0 0 26397-128-4 C:/WINDOWS/Installer/{49FD463C-18F1-63C4-8F12-49F518F127}/t111o00000000.dat
Thu Jun 14 2012 12:55:12 285691 mac. r/rrwxrwxrwx 0 0 26397-128-4
C:/WINDOWS/Installer/{49FD463C-18F1-63C4-8F12-49F518F127}/t111o00000000.dat
4096 ..c. -/rr-xr-xr-x 0 0 26447-128-4

The infected process “winlogon.exe” was observed writing this file via Process:



winlogon.exe	420	CreateFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	SetEndOfFileInformationFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	SetAllocationInformationFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat
winlogon.exe	420	WriteFile	C:\WINDOWS\Installer\{49FD463C-18F1-63C4-8F12-49F518F127}\t111o00000000.dat

Examination of this file reveals that it is a screenshot of the desktop:



Many other modules providing specific exfiltration capabilities were observed. Generally, the exfiltration modules write files to disk using the following naming convention: XXY1TTTTTTTT.dat. XX is a two-digit hexadecimal module number, Y is a single-digit hexadecimal submodule number, and TTTTTTTT is a hexadecimal representation of a unix timestamp (less 1.3 billion) associated with the file creation time.

ENCRYPTION

The malware uses encryption in an attempt to disguise harvested data in the .dat files intended for exfiltration. Data written to the files is encrypted using AES-256-CBC (with no padding). The 32-byte key consists of 8 readings from memory address 0x7ffe0014: a special address in Windows that contains the low-order-4-bytes of the number of hundred-nanoseconds since 1 January 1601. The IV consists of 4 additional readings.

The AES key structure is highly predictable, as the quantum for updating the system clock **(HKLM\SYSTEM\CurrentControlSet\Services\W32Time\Config\LastClockRate)** is set to

0x2625A hundred-nanoseconds by default, and the clock readings that comprise the key and IV are taken in a tight loop:

```
...
0x406EA4: 8D45C0 LEA EAX,[EBP-0x40]
0x406EA7: 50 PUSH EAX
0x406EA8: FF150C10AF01 CALL DWORD PTR [0x1AF100C]
0x406EAE: 8B4DE8 MOV ECX,DWORD PTR [EBP-0x18]
0x406EB1: 8B45C0 MOV EAX,DWORD PTR [EBP-0x40]
0x406EB4: 8345E804 ADD DWORD PTR [EBP-0x18],0x4
0x406EB8: 6A01 PUSH 0x1
0x406EBA: 89040F MOV DWORD PTR [EDI+ECX],EAX
0x406EBD: FF152810AF01 CALL DWORD PTR [0x1AF1028]
0x406EC3: 817DE800010000 CMP DWORD PTR [EBP-0x18],0x100
0x406ECA: 72D8 JB 0x406EA4
0x406ECC: 80277F AND BYTE PTR [EDI],0x7F
...
```

The following AES keys were among those found to be used to encrypt records in .dat files. The first contains the same 4 bytes repeated, whereas in the second key, the difference between all consecutive 4-byte blocks (with byte order swapped) is 0x2625A.

```
70 31 bd cc 70 31 bd cc 70 31 bd cc 70 31 bd cc 70 31 bd cc 70 31
bd cc 70 31 bd cc

26 e9 23 60 80 4b 26 60 da ad 28 60 34 10 2b 60 8e 72 2d 60 e8 d4 2f 60 42 37
32 60 9c 99 34 60
```

In all, 64 clock readings are taken. The readings are encrypted using an RSA public key found in memory (whose modulus begins with A25A944E) and written to the .dat file before any other encrypted data. No padding is used in the encryption, yielding exactly 256 encrypted bytes. After the encrypted timestamp values, the file contains a number of records encrypted with AES, delimited by EAE9E8FF.

In reality, these records are only partially encrypted: if the record's length is not a multiple of 16 bytes (the AES block size), then the remainder of the bytes are written to the file unencrypted. For example, after typing "FinSpy" on the keyboard, the keylogger module produced the following (trailing plaintext highlighted):

```
00000200  ed ff c5 7e 0e 8e 17 4b  33 80 2f 9a 74 92 b6 50  |...~...K3./.t..P|
00000210  41 ba fc 1d 7f ce ff 52  cf 68 1f d1 ea 8a 3b 5d  |A......R.h....;]|
00000220  b5 1a fe eb eb 54 e2 4a  12 d1 24 33 60 cd 2e f6  |.....T.J..$3`...|
00000230  da dc 86 6a 56 c6 df 6d  b5 18 5c 96 14 a3 84 13  |...jV..m..\.....|
00000240  3e 27 25 dd 33 72 56 e8  be 5c e5 54 3a dc 96 e2  |>'%.3rV..\.T:...|
00000250  4f cc 3f e9 16 76 8b 6e  bf 61 73 40 2e 15 11 d7  |O.?..v.n.as@....|
00000260  73 a1 c6 12 c2 c6 7f 56  08 bb 37 50 5f 55 54 99  |s......V..7P_UT.|
00000270  d3 21 2c 59 2a 27 48 01  54 b5 45 a7 d7 b5 32 62  |.!,Y*'H.T.E...2b|
00000280  dd 15 fc 46 00 00 00 90  03 fe 00 ea e9 e8 ff 38  |...F...........8|
00000290  01 3a 64 e2 98 58 c7 e6  b7 96 7f 68 8d 1f 4e 09  |.:d..X.....h..N.|
000002a0  b1 9f 29 7f e4 dd e2 9f  b9 4b eb 3d 4b 4a 8b 42  |..).......K.=KJ.B|
000002b0  81 b5 6a 76 db d8 1c 36  ad a9 25 1f 40 b5 ef 69  |..jv...6..%.@..i|
000002c0  00 6e 00 53 00 70 00 79  00                       |.n.S.p.y.|
```

The predictability of the AES encryption keys allowed us to decrypt and view these partially-encrypted records in full plaintext. The nature of the records depends on the particular module and submodule. For example, submodule Y == 5 of the Skype exfiltration module (XX == 14), contains a csv representation of the user's contact list:

Record # 0 Length: 243 bytes:
ó
@þÿÌ³Ð
@
¤b¯Opþ192.168.131.67JRecordingEcsv 0þ-0800UTC DST.1þ2012-07-18 18:00:21.:þ1970-01-01 00:16:00Abhwatch1

Record # 1 Length: 96 bytes:
`USERNAME,FULLNAME,COUNTRY,AUTHORIZED,BLOCKED

Record # 2 Length: 90 bytes:
Zecho123,Echo / Sound Test Service,,YES,NO

Record # 3 Length: 95 bytes:
^bhwatch2,Bahrain Watch,United States,YES,NO

Submodule Y == 3 records file transfers. After a Skype file transfer concludes, the following file is created: %USERPROFILE%\Local Settings\Temp\smtXX.tmp. This file appears to contain the sent / received file. As soon as smtXX.tmp is finished being written to disk, a file (1431XXXXXXXX.dat) is written, roughly the same size as smtXX.tmp. After sending a picture (of birdshot shotgun shell casings used by Bahrain's police) to an infected Skype client, the file 1431028D41FD.dat was observed being written to disk. Decrypting it revealed the following:

Record # 0 Length: 441 bytes:

1

@þÿÌ³Ð

@

¤b¯Opþ192.168.131.67Abhwatch1Bbhwatch2"CBahrain WatchIreceivedrC:\Documents and Settings\XPMUser\My Documents\gameborev3.jpgJRecording 0þ-0800UTC DST.1þ2012-07-20 12:18:21.:þ2012-07-20 12:18:21

Record # 1 Length: 78247 bytes:

[Note: Record #1 contained the contents of the .jpg file, preceded by hex A731010090051400, and followed by hex 0A0A0A0A.]

Additionally, submodule Y == 1 records Skype chat messages, and submodule Y == 2 records audio from all participants in a Skype call. The call recording functionality appears to be provided by hooking DirectSoundCaptureCreate:

```
************************************************************************
Hook mode: Usermode
Hook type: Inline/Trampoline
Process: 424 (winlogon.exe)
Victim module: dsound.dll (0x73f10000 - 0x73f6c000)
Function: dsound.dll!DirectSoundCreate at 0x73f1473b
Hook address: 0x2943b1a
Hooking module: <unknown>

Disassembly(0):
0x73f1473b e9daf3a28e       JMP 0x2943b1a
0x73f14740 51               PUSH ECX
0x73f14741 8b0d0460f673     MOV ECX, [0x73f66004]
0x73f14747 8365fc00         AND DWORD [EBP-0x4], 0x0
0x73f1474b 56               PUSH ESI
0x73f1474c 57               PUSH EDI
0x73f1474d e8b9d6ffff       CALL 0x73f11e0b
0x73f14752 83               DB 0x83

Disassembly(1):
0x2943b1a 8bff             MOV EDI, EDI
0x2943b1c 55               PUSH EBP
0x2943b1d 8bec             MOV EBP, ESP
0x2943b1f 56               PUSH ESI
0x2943b20 ff7510           PUSH DWORD [EBP+0x10]
0x2943b23 8b750c           MOV ESI, [EBP+0xc]
0x2943b26 56               PUSH ESI
0x2943b27 ff7508           PUSH DWORD [EBP+0x8]
0x2943b2a ff15c4ac9402     CALL DWORD [0x294acc4]
0x2943b30 85c0             TEST EAX, EAX

************************************************************************
```

COMMAND AND CONTROL

This section describes the communications behavior of the malware.

When we examined the malware samples we found that they connect to a server at IP address 77.69.140.194

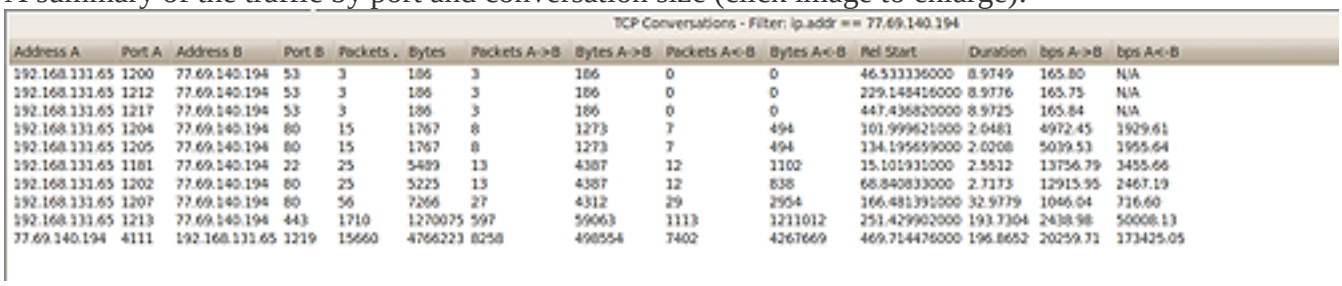
PM	iexplore.exe	1908	TCP Send	:1181 -> static.ip.77.69.140.194.batelco.com.bh:22
PM	iexplore.exe	1908	TCP Send	:1181 -> static.ip.77.69.140.194.batelco.com.bh:22
PM	iexplore.exe	1908	TCP Receive	:1181 -> static.ip.77.69.140.194.batelco.com.bh:22
PM	iexplore.exe	1908	TCP Disconnect	:1181 -> static.ip.77.69.140.194.batelco.com.bh:22
PM	iexplore.exe	1908	TCP Reconnect	:1200 -> static.ip.77.69.140.194.batelco.com.bh:domain
PM	iexplore.exe	1908	TCP Reconnect	:1200 -> static.ip.77.69.140.194.batelco.com.bh:domain
PM	iexplore.exe	1908	TCP Disconnect	:1200 -> static.ip.77.69.140.194.batelco.com.bh:domain
PM	iexplore.exe	1908	TCP Send	:1202 -> static.ip.77.69.140.194.batelco.com.bh:http
PM	iexplore.exe	1908	TCP Send	:1202 -> static.ip.77.69.140.194.batelco.com.bh:http
PM	iexplore.exe	1908	TCP Receive	:1202 -> static.ip.77.69.140.194.batelco.com.bh:http

WHOIS data[7] reveals that this address is owned by Batelco, the principal telecommunications company of Bahrain:

```
inetnum: 77.69.128.0 - 77.69.159.255
netname: ADSL
descr: Batelco ADSL service
country: bh
```

For a period of close to 10 minutes, traffic was observed between the infected victim and the command and control host in Bahrain.

A summary of the traffic by port and conversation size (click image to enlarge):



TCP Conversations - Filter: ip.addr == 77.69.140.194

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A->B	Bytes A->B	Packets A<-B	Bytes A<-B	Rel Start	Duration	bps A->B	bps A<-B
192.168.131.65	1200	77.69.140.194	53	3	186	3	186	0	0	46.533336000	8.9749	165.80	N/A
192.168.131.65	1212	77.69.140.194	53	3	186	3	186	0	0	229.148416000	8.9776	165.75	N/A
192.168.131.65	1217	77.69.140.194	53	3	186	3	186	0	0	447.436820000	8.9725	165.84	N/A
192.168.131.65	1204	77.69.140.194	80	15	1767	8	1273	7	494	101.999621000	2.0481	4972.45	1929.61
192.168.131.65	1205	77.69.140.194	80	15	1767	8	1273	7	494	134.195659000	2.0208	5039.53	1955.64
192.168.131.65	1181	77.69.140.194	22	25	5489	13	4387	12	1102	15.101931000	2.5512	13756.79	3455.66
192.168.131.65	1202	77.69.140.194	80	25	5225	13	4387	12	838	68.840833000	2.7173	12915.95	2467.19
192.168.131.65	1207	77.69.140.194	80	56	7266	27	4312	29	2954	166.481391000	32.9779	1046.04	716.60
192.168.131.65	1213	77.69.140.194	443	1710	1270075	597	59063	1113	1211012	251.429902000	193.7304	2438.98	50008.13
77.69.140.194	4111	192.168.131.65	1219	15660	4766223	8258	498554	7402	4267669	469.714476000	196.8652	20259.71	173425.05

The infected VM talks to the remote host on the following five TCP ports:

22
53
80
443
4111

Based on observation of an infected machine we were able to determine that the majority of data is exfiltrated to the remote host via ports 443 and 4111.

192.168.131.65:1213 -> 77.69.140.194:443 1270075 bytes
192.168.131.65:4111 -> 77.69.149.194:4111 4766223 bytes

CONCLUSIONS ABOUT MALWARE IDENTIFICATION

Our analysis yields indicators about the identity of the malware we have analyzed: (1) debug strings found the in memory of infected processes appear to identify the product and (2) the samples have similarities with malware that communicates with domains belonging to Gamma International.

Debug Strings found in memory

As we previously noted, infected processes were found containing strings that include "finspyv4.01" and "finspyv2":

y:\lsvn_branches\finspyv4.01\finspyv2\src\libs\libgmp\mpn-tdiv_qr.c
y:\lsvn_branches\finspyv4.01\finspyv2\src\libs\libgmp\mpn-mul_fft.c
y:\lsvn_branches\finspyv4.01\finspyv2\src\target\bootkit_x32driver\objfre_w2k_x86\i386\bootkit_x32driver.pdb

Publicly available descriptions of the FinSpy tool collected by Privacy International among others and posted on Wikileaks[8] make the a series of claims about functionality:

- Bypassing of 40 regularly tested Antivirus Systems
- Covert Communication with Headquarters
- Full Skype Monitoring (Calls, Chats, File Transfers, Video, Contact List)
- Recording of common communication like Email, Chats and Voice-over-IP
- Live Surveillance through Webcam and Microphone
- Country Tracing of Target
- Silent Extracting of Files from Hard-Disk
- Process-based Key-logger for faster analysis
- Live Remote Forensics on Target System
- Advanced Filters to record only important information
- Supports most common Operating Systems (Windows, Mac OSX and Linux)

Shared behavior with a sample that communicates with Gamma

The virtual machine used by the packer has very special sequences in order to execute the virtualised code, for example:

```
66 C7 07 9D 61 mov word ptr [edi], 619Dh
C6 47 02 68 mov byte ptr [edi+2], 68h
89 57 03 mov [edi+3], edx
C7 47 07 68 00 00 00 mov dword ptr [edi+7], 68h
89 47 08 mov [edi+8], eax
C6 47 0C C3 mov byte ptr [edi+0Ch], 0C3h
```

Based on this we created a signature from the Bahrani malware, which we shared with another security researcher who identified a sample that shared similar virtualised obfuscation. That sample is:

md5: c488a8aaef0df577efdf1b501611ec20
sha1: 5ea6ae50063da8354e8500d02d0621f643827346
sha256: 81531ce5a248aead7cda76dd300f303dafe6f1b7a4c953ca4d7a9a27b5cd6cdf

The sample connects to the following domains:

tiger.gamma-international.de
ff-demo.blogdns.org

The domain **tiger.gamma-international.de** has the following Whois information[9]:

```
Domain: gamma-international.de

Name: Martin Muench
Organisation: Gamma International GmbH
Address: Baierbrunner Str. 15
PostalCode: 81379
City: Munich
CountryCode: DE
Phone: +49-89-2420918-0
Fax: +49-89-2420918-1
Email: info@gamma-international.de
Changed: 2011-04-04T11:24:20+02:00
```

Martin Muench is a representative of Gamma International, a company that sells "advanced technical surveillance and monitoring solutions". One of the services they provide is FinFisher: IT Intrusion, including the FinSpy tool. This labelling indicates that the matching sample we were provided may be a demo copy a FinFisher product per the domain **ff-demo.blogdns.org**.

We have linked a set of novel virtualised code obfuscation techniques in our Bahraini samples to another binary that communicates with Gamma International IP addresses. Taken alongside the explicit use of the name "FinSpy" in debug strings found in infected processes, we suspect that the malware is the FinSpy remote intrusion tool. This evidence appears to be consistent with the theory that the dissidents in Bahrain who received these e-mails were targeted with the FinSpy tool, configured to exfiltrate their harvested information to servers in Bahraini IP space. If this is not the case, we invite Gamma International to explain.

RECOMMENDATIONS

The samples from email attachments have been shared with selected individuals within the security community, and we strongly urge antivirus companies and security researchers to continue where we have left off.

Be wary of opening unsolicited attachments received via email, skype or any other communications mechanism. If you believe that you are being targeted it pays to be especially cautious when downloading files over the Internet, even from links that are purportedly sent by friends.

ACKNOWLEDGEMENTS

Malware analysis by Morgan Marquis-Boire and Bill Marczak. Assistance from Seth Hardy and Harry Tuttle gratefully received.

Special thanks to John Scott-Railton.

Thanks to Marcia Hofmann and the Electronic Frontier Foundation (EFF).

We would also like to acknowledge Privacy International for their continued work and graciously provided background information on Gamma International.

FOOTNOTES

[1] http://www.finfisher.com/
[2] http://owni.eu/2011/12/15/finfisher-for-all-your-intrusive-surveillance-needs/#SpyFiles
[3] http://blogs.aljazeera.com/profile/melissa-chan
[4] This technique was used in the recent Madi malware attacks.
[5] http://www.finfisher.com/
[6] Unpacking Virtualised Obfuscators by Rolf Rolles - http://static.usenix.org/event/woot09/tech/full_papers/rolles.pdf
[7] http://whois.domaintools.com/77.69.140.194
[8] E.g. http://wikileaks.org/spyfiles/files/0/289_GAMMA-201110-FinSpy.pdf
[9] http://whois.domaintools.com/gamma-international.de

Back to top

MEDIA COVERAGE

The Wall Street Journal
Slate
CSO
Tech Week Europe
Bloomberg
Electronic Frontier Foundation
Privacy International
Spiegel Online
PC Mag
The New York Times

About the Author

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.