“Red October”. Detailed Malware Description 3. Second Stage of Attack

securelist.com/red-october-detailed-malware-description-3-second-stage-of-attack/36802/

By GReAT

First stage of attack

1. Exploits
2. Dropper
3. Loader Module
4. Main component

Second stage of attack

1. Modules, general overview
2. Recon group
3. Password group
4. Email group
5. USB drive group
6. Keyboard group
7. Persistence group
8. Spreading group
9. Mobile group
10. Exfiltration group

3. Password group

PswSuperMailRu module

Known variants:

MD5	Size	Compilation date (payload)
7b669c32e6ee2c65bec5e09024fc5415	237,568	January 14th, 2011
b7327bfa4a101a21f0cc1b366aa8e107	237,568	March 29th, 2011
a39fa7340b2f1d7b42342b3e2f06df71	266,24	August 16th, 2010
cd170625655424149573c88c59716cc4	270,336	February 11, 2011
f60436b984962e741b81720ea604ad27	241,664	August 16th, 2010
2cf23cd8a7f85529576ba6c759f8cf37	270,336	March 29th, 2011
9bb32272be87a4dde8c8b05f49ded9f7	266,24	January 14th, 2011
ed72b6150e9fbc8f71e61dfea682a303	237,568	February 11th, 2011

The files are all PE exe files between 232 KB and 264 KB, compiled with Microsoft Visual Studio 2005. It is written in C/C++.

Entrypoint, unpacking stub, and obfuscation

Related binaries are heavily packed with a custom packer. The packer disrupts basic software breakpoints and some api hooking techniques, because it decrypts the original exe's section contents onto heaps in-memory and then writes the decrypted code and .data, .rdata contents back over the original sections, hopping back into the decrypted code and executing from there.

Summary

At its OEP, the module further retrieves environment information, and then creates a mutex with the string "SUPER MUTEX". Then the module steals authentication credentials from mail.ru agent software and attempts to steal open and saved attachments on the system from the Outlook Secure Temp Folder. The executable then fails to write its own contents to another executable file in tmp and maintain persistence on the system. Network functionality is absent from this module.

Credential Stealing

The module steals Mail.Ru Mail Agent account credentials by reading relevant registry in order to extract the passwords or decrypt them depending of the version. The two locations this data is accessed are

- Registry key/value pair for **Mail.Ru Agent version** < 5.2
  Registry key: HKCUSoftwareMail.RuAgentmagent_logins,
  binary value: ####password
- Registry key/value pair for **Mail.Ru Agent version 5.2 to 5.6**
  Registry key: HKCUSoftwareMail.RuAgentmagent_logins2
  binary value: ####password
- Registry key/value pair for **Mail.Ru Agent version >= 5.7** (Last version 6 is also supported)
  Registry key: HKCUSoftwareMail.RuAgentmagent_logins3
  binary value: ####password
- Text file used by Mail.Ru Agent to store Blowfish encryption key %APPDATA%MraUpdatever.txtThe contents of this registry keys are read and the blowfish key stored inside "ver.txt" is used to decrypt the passwords, in version 5.7 and above.All actions are logged with time stamp, and then encrypted and written to %temp%~avp2.log.The module will attempt to access "ver.txt" which is protected on the system. If it can read it using standard API functions, it uses **NTFS low level access**.In order to do so, it uses the DeviceIoControl API function with the "FSCTL_GET_NTFS_VOLUME_DATA" and "FSCTL_GET_NTFS_FILE_RECORD" IOCTLs.The data of this file will also be written to %temp%~mslog.tmp, which is a backup of "ver.txt"After the module will build a new string with the current timestamp followed by delimiting dashes and indication that the module has finished running, like "2012:11:10:10:22:34——-PROGA END". This string will be written to the

~avp2.log file.Here is an example of a decrypted ~avp2.log file:
“2012:11:22:17:21:44”——-PROGA START——
“2012:11:22:17:21:44"ERROR ACCESS file ver.txt
“2012:11:22:17:21:44"FILE SYSTEM NTFS
“2012:11:22:17:21:44"PROBING GET FILE LOW LIVEL
“2012:11:22:17:21:44"ACCESS LOW LIVEL OK!!!!
“2012:11:22:17:21:44"REG OPEN KEY[SoftwareMail.RuAgentmagent_logins3]
“2012:11:22:17:21:44"SUB KEY[000#victim@mail.ru] “2012:11:22:17:21:44"uin(0)
[000#victim@mail.ru] passwd(0)[%removed%] “2012:11:22:17:21:44”——-PROGA END
——
The string between “[” and “]” is binary value of MD5 Hash of the Mail.ru Agent password.All restored credentials are also stored encrypted in the “~pass2.pwl” file without the time stamp information and how they were accessed.Meanwhile “PROGA” word used in here might refer to transliteration of Russian slang “ПРО”which literally means an application or a program.

Attachment stealing

The module will finally attempt to steal open attachments or attachments that were open when the machine or Outlook application shutdown unexpectedly. It identifies and collects these files from a directory path that looks like “C:\Documents and Settings\%username%\Local Settings\Temporary Internet Files\OLK%xxx%“.

Failed persistence

The module then attempts to maintain some persistence on the system by repeating the routine. It creates %temp%\19d400.msv, and attempts to write a copy of its own executable contents to it. Oddly, the WriteFile size parameter is set to “0” and the write fails. It then attempts to run this 19d400.msv file by calling CreateProcessA on it, which also fails. The module thread terminates the process.

Finally, it is interesting that network functionality is absent from this module.

Interesting decrypted runtime strings:

19d400.msv
Software\Mail.Ru\Agent\magent_logins2
Software\Mail.Ru\Agent\magent_logins
~pass.pwl
~avp.log
www.mail.ru
ProcessRead ok!
OPEN PROCESS agent_mail.ru OK!
%s——-PROGA END——
——-PROGA START——
SUPER MUTEX
19d400.msv
~mslog.tmp
%sPROBING GET FILE LOW LIVEL
%sACCESS LOW LIVEL OK!!!!
%sERROR OPENFILE

%sERROR GET ACCESS LOW LIVEL

PswOutlook module

Known variants:

MD5	Compilation date
f6e1637e04b33a3e0c57ab355d3e677e	2010.11.30 07:10:32 (GMT)
fa66821fd895b3814e501b804176ef98	2011.02.23 12:41:29 (GMT)

Summary

The file is a PE EXE file, compiled with Microsoft Visual Studio 2008. All the functionality is implemented in the WinMain function. There are 2 known variants of this module in over 50 files with identical behavior.

Like PswSuperMailru, this highly-obfuscated module is very different from others. Its main purpose is to steal email credential information of the current user. This is achieved by reading system protected storage and system registry. The result is stored in an encrypted file, after that the application self-removes.

Main function

This module starts from decrypting pieces of information carefully puzzled in the file body. The puzzle contains extra library names, export function names, their parameters and set of internal references. The encryption algorithm reminds PKZIP encryption but seems to be modified.

Decrypted data is carefully collected and filled in a set of internal objects. Next, module connects to the local registry using **RegConnectRegistry** system API call. This is not clear why the developers decided to use RegConnectRegistry call. They either tried to bypass some local IDS/IPS systems by avoiding usage of RegOpenKey/RegOpenKeyEx calls or the application was designed to connect to remote computers as well. In all samples we have observed, **lpMachineName** parameter is set to NULL meaning to connect to local registry.

It dumps MS Outlook account information from the following registry keys:

HKCU\Software\Microsoft\Internet Account Manager
HKCU\Software\Microsoft\Office\Outlook\OMI Account Manager\Accounts
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging
HKCU\Subsystem\Profiles\Microsoft\Outlook Internet Settings
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook

After that it attempts to access and dump local protected storage identities information via **PStorec.dll**API.

Collected information is stored in encrypted file: %Temp%\{B30BD41D-46E7-458B-F412-4D7F80CCAD0F}. The file contains a 32bit hash of data in the end.

This module never communicates with the C&C server and works as a standalone tool. In the end of execution before exiting application it runs self-removal procedure as described below.

Self-removal procedure

When the module needs to end the execution and self-remove, it spawns a %COMSPEC% process (cmd.exe or command.com on older systems) HKCU\Software\Microsoft\Internet Account Managerusing the following commandline:

cmd.exe /c del /F /A "%MODULE_PATH%">> NULL

This procedure is very unreliable as it is prone to race-condition issues which might result in modules remaining undeleted.

MSHash module

Known variants:

MD5	Compilation date
3538fea2c2f9a7117a6a919c87112731	2011.11.01 15:04:58 (GMT)
a008d1ec659c3758e95bc3f0aafbe3a5	2011.08.05 07:40:37 (GMT)
68d72e12c402038195175b568b3dd0bb	2012.10.22 07:01:30 (GMT)
4b62cc78508b46d74cdd172dc493ec8a	2011.11.01 15:04:58 (GMT)

Summary

This module is a standalone executable, which is essentially a tool to dump cached domain password hashes, and locally stored sensitive information, such as LSA secrets. It uses direct disk access to bypass system registry ACLs. After execution the module self-removes.

First, it uses direct disk access to copy registry files from %SYSTEM%\config\ directory:

%SYSTEM%\config\sam -> %Temp%\ksm
%SYSTEM%\config\system -> %Temp\kss
%SYSTEM%\config\SECURITY -> %Temp%\kse

Then module fetches available cached domain account hashes as well as local system LSA secrets. The later may contain logins and passwords from various services in plaintext. Also, it fetches Syskey bootkey secret and appends it to the output.

The result is stored in an encrypted file of custom binary format which is located in%TMP%\smrdprev.tmp. The contents of the file contains internal file reference string, including date and time when it was created, i.e. "@MSHASH\SAMHASH_2_20121002_034519.txt". This is probably suggested relative path and file name during data extraction procedure on the attackers system.

The module also creates an encrypted log file with detailed information of program execution. It is stored in similar file “%TMP%\smrdprev.tmp“, the hex number depends on current system boot time. It is encrypted in the same way as the main output file, using custom cryptoalgorithm based on AMPRNG cryptomethod.

Current module also tags current system by changing the following registry keys:

HKCU\Software\Microsoft\ADOSoftware32\ProductID = binary value of 20 bytes (System ID)
HKLM\Software\Microsoft\ADOSoftware32\ProductID = binary value of 20 bytes (System ID)

System ID is calculated as SHA1-hash of System Drive Volume Serial Number and HKLM\SOFTWARE\Microsoft\Internet Explorer\Registration\ProductID value. These keys remain after malware self-removes an may serve as good way to identify if your system was hit by this module in the past.

Most of the cryptoroutines such as DES, RC4, HMAC_MD5, MD4 are used from statically linked OpenSSL 0.9.8g library.

After the end of execution it deletes temporary files and self-removes with simple msc.bat file of the following contents:

```
chcp 1251
:Repeat
attrib -a -s -h -r “”
del “”
if exist “” goto Repeat
del “msc.bat”
```

Note, that current batch file sets current codepage to CP1251, which is used to display Cyrillic characters in console output.

4. Email group

MAPIClient module

Known variants:

MD5	Compilation date
09e75477e03a968eead17a28d8aef0ce	2012.10.26 07:02:24 (GMT)
10603f7ec89c3472b238e9342f5ba62b	2011.10.10 11:37:27 (GMT)
C196e32764dc698bb88714adfb874667	2012.05.04 11:31:35 (GMT)
0fe600e06a69ccebbb5baf6c9f5f51a6	2011.12.02 07:34:41 (GMT)
c3a50d78669cd58b2cd4e38e30c1e986	2011.11.11 07:13:55 (GMT)
298c4562c8463bed3039ff2d12669adc	2011.09.02 05:08:00 (GMT)
1f91b25d0893d4e1b0418ffeb21f1f03	2011.10.10 11:37:27 (GMT)

521b45d21b4b2fc48f7ab29ab222d6ee	2011.11.11 07:13:55 (GMT)
7883b174ce69ffed41d3aea54855ff97	2011.06.10 06:11:48 (GMT)
3975b42d9bb39741e988f78020edeb44	2011.11.11 07:13:55 (GMT)

Summary

The file is a PE EXE file, compiled with Microsoft Visual Studio 2010. All the functionality is implemented in the WinMain function.

This module is used to steal email information of the current user by getting most valuable information about messages, starting from general fields (To/From/Date/Subj), copying full MIME headers and message body, and stealing attachments if extension looks interesting (documents, archives, cryptokeys). The result is stored in a set of encrypted and compressed files, after that the application self-removes.

Main function

It starts from creating a new encrypted log file in current directory of the executable, named “system32ocxms.dat”. The first log entries contain the following:

Program started
V: MSG_26.10.12 This probably indicate internal module name (MSG) and version (set after date of creation 26.10.12). Current application hasn’t got an embedded config/script file in resource section, all parameters are hardcoded.

Next it fetches the current system general information and default application preferences from the registry, including the following:

User name
Computer name
User domain name (from %USERDOMAIN% and %USERDNSDOMAIN%)
Default system HTTP protocol handler from HKCR\HTTP\shell\open\command
Default system HTTPS protocol handler from HKCR\https\shell\open\command
Default system application for html files from HKCR\htmlfile\shell\open\command
Default system e-mail client from HKCR\mailto\shell\open\command

This information is added to the log file.

Next it checks if MS Outlook is installed, if it is actively running, whether it is system default e-mail client. This is appended to the log either.

During next step, it creates a watchdog thread, which every checks foreground window every 10 milliseconds and in case current foreground window belongs to current process it spawns self-removal batch-file (see description in section below) and terminates current process. This is done in case Microsoft Windows MAPI framework creates a popup message with some connection error report, or asks the user to enter credentials from the mailbox. The developers obviously wanted to

escape unnecessary attention and that is why they thought that it would be better to shutdown and self-remove the module without collecting any information instead of getting caught by attracting attention.

After watchdog thread was created the module lists local user mail addresses by connecting to system MAPI. It gets User Email, Address Type and User Name. Next it iterates through existing MAPI storages for current user and dumps full information about all messages to the main log file, including e-mail header values, such as

Subject
From
To
Date
AttachmentID
AttachmentName
AttachmentSize

The emails of attackers' interest with attachments are saved in a separate encrypted and Zlib-compressed storage file (new storage file is created for each directory):

%CurrentDir%\sys_%HexNumber%\winocx_%IntegerCounter%_rdtp.tmp

The attachment storage file is limited to 60Mb, RC4 encryption key is

```
1  specialforoutlookattachments!
```

The filter for interesting emails is defined in a list of regular expressions (PCRE lib is used to process those):

.doc .xls .txt .vcf .p7m .mdb .msgz .dcx .egk .zm9 .docx .xlsx .eml .CCC .cif .fas .msg .kum .key .out .rtf .pdf .enc .TOK .wps .sbox .lock .ass .cer .pem .odt .rar .html .xia .sec .cm5 .xps .pgp .wfh .cbu .ods .zip .htm .xis .dat .rhs .x05 .xiti .grp .sig .dot .gpg mht .p7s .dth .max .y05 .egm .pot .ftil

Current module is also capable of dumping full contents of local system **Address Book**, however this functionality is currently disabled by the developer (probably because of some bugs in the code).

This module never communicates with the C&C server and works as a standalone tool. In the end of execution before exiting application it runs self-removal procedure as described below.

Self-removal procedure

When the module needs to end the execution and self-remove, it creates a batch-file with pseudo-random name: %HOMEDRIVE%%HOMEPATH%\Local Settings\Temp\.bat.

If it couldn't create random name, the name will be set to "syspart.bat".

The file contains:

:Repeat
del /F /A “”
if exist “” goto Repeat
del /F /A “”

POP3Client module

Known variants:

MD5	Compilation date (encrypted)	Compilation date (payload)
224c382316be4be7e0009f08b84cd91e	2011.09.26 06:54:09 (GMT)	2011.05.25 11:49:19 (GMT)
100e53ee8fbeb4546b31eb7e0aad8752	2011.07.27 07:37:01 (GMT)	2011.05.25 11:49:19 (GMT)

Summary

The file is a PE EXE file, compiled with Microsoft Visual Studio 2005.

This module is responsible for receiving and storing email messages on a local computer from a POP3 server specified in a configuration file. All the actions and important info are written to a log file.

Main function

Creates mutex “208D2C60-3AEA-1069-A2D7-08002B30309D”

Creates a directory and a log file in it: %ALLUSERSPROFILE%\Application Data\System\smrdprev.%d%d.tmp (%d values correspond to time64() ^ 0x1F3E231 and GetTickCount()).

Installs in system AutoRun:

>HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
“LgfxTray”= PathToExe

Other Registry entries:

HKLM\SOFTWARE\Microsoft\ADOSoftware32
HKCU\SOFTWARE\Microsoft\ADOSoftware32
“ProductID” = SHA1(MAC, AdapterIP, VolumeSerialNumber, IE Product ID)

Always starts the log file from:

@LOG\CMAIL_LOCAL_v1_ %YEAR%%MONTH%%DAY%_%HOUR%%MIN%%SEC%.txt Software version: 0.8
Current Directory: %s
“———PROGA START——-”

Opens mutex “huiofwhfiowjcpowjkcwcophwvurweionwopmcvopwkvpwjnhopv”, if it exists, then module terminates.

Tries to open config file “jusched32s.dat” and read the contents. If the file doesn’t exist, then terminates. Checks file MD5 hash in file header (first 16 bytes), if it was changed after the last saved MD5 (in memory as a variable), then updates it and writes the new hash to log file.

If calculated hash of the file buffer matched the header hash then module decrypts and parses config file and launches POP3 client thread, otherwise terminates.

Parameters in config file:

common
account
config_id
email_hash_path
storage_path
storage_size
load_period
time_delta
size_min
size_max
storage_hash_size
uninstall
login
password
pop3server
imapserver
use
email

If “unistall” parameter is set in config file, then module deletes AutoRun keys in registry, config filejusched32s.dat and terminates.

POP3 client thread

By default all connections are not encrypted with SSL. There is a variable which for some reason is not initialized, that is used to control the network communication, i.e. to use SSL encryption or not.

Establishes a connection based on ‘pop3server’ (or ‘imapserver’), ‘login’ and ‘password’ parameter values in config file.

After receiving the number of emails as a response on STAT command starts processing emails in cycle.

Forms an MD5 hash from a string in format “%s%s%s” with ‘login’ parameter, ‘pop3server’ parameter and the value that depends on current processed email number and UIDL response.

Seeks in directory specified in ’email_hash_path’ for *mso.dat files, reads their contents and compares the hashes stored in them with calculated hash.

If calculated hash is found in the contents, then the module doesn't process the current email, otherwise gets email message size from a POP3 server with a LIST command.

Creates File %email_hash_path%\%s%.mso.dat (%s – CRC32(GetUserName)) and appends counted hash to it. If the new File Size is bigger than 'storage_hash_size' than rewrites the file with the latest data so it's size doesn't exceed 'storage_hash_size'.

If email size is in the range 'size_min'-'size_max' then the module retrieves email headers and email message body, otherwise stops email processing. Then the module parses the response, retrieves the date of email, counts days elapsed and compares it to 'time_delta'. If it's bigger than 'time_delta' then breaks the email processing cycle.

Creates file: %storage_path%\bcmntc.%d.tmp (%d – time64 ^ 0x1F3E231.tmp) with decrypted contents including email header: @CMAIL_LOCAL\%s_%04u%02u%02u_%02u%02u%02u\%d.eml ('email' field, Date, Email number) and email contents itself. Some buffers are compressed with zlib.

All the collected data is stored locally and isn't uploaded to any C&C by this module.

5. USB Drive group

USBContainer module

Known files

MD5	Compilation date (encrypted)
b9568a91d6f6b0904de8b2e9d9a2d32c	2010.06.01 11:24:07 (GMT)
f0eaec0b25afc24a416810fe46242590	2010.06.01 11:24:07 (GMT)
865ba7958efe7e54501dcf2c19dcd99e	2010.06.01 11:24:07 (GMT)

Summary

This is a standalone EXE application module which is used to drop and run USBStealer module (IGFXTRAYMS.exe) along with its configuration file (imapisync32.dat)

Those two files are zlib compressed and stored in the overlay of the dropper.

Main function

Upon execution, the dropper reads its overlay and decompresses it in memory.

The configuration file "imapisync32.dat" is dropped first followed by the opening of a system event named "ScxinWordSid_0129211FA". This event is created by the USBStealer module.

Afterwards, it will try to delete the “IGFXTRAYMS.EXE” file without checking if it exists or not.

The USBStealer module is then dropped using the following file name: “IGFXTRAYMS.EXE” and executed. Both files are dropped in the same directory of the dropper.

Finally, the following will be executed 3 times before exiting:

C:\WINDOWS\system32\cmd.exe /c del C:\DOCUME~1\%PATH_TO_DROPPER.EXE% >> NUL

This module doesn’t create any execution log files, nor does it connect to the C&C servers.

USBRestore

Known files

MD5	Compilation date (encrypted)
9572cc04fd442027cfd61178bdf73c0c	2011.07.15 12:30:29 (GMT)
feba0bbead1a810c223cf8252b529d65	2011.06.02 12:30:50 (GMT)
4aabfd510ef66e066946087617638090	2011.06.02 12:30:50 (GMT)
1d124d06666cfa6b33768f1147208b9c	2011.07.15 12:30:29 (GMT)
260ad160972ca6bc071b7cb518a9b5fa	2011.06.02 12:30:50 (GMT)
ab72d7ed99c3c18f2582b6e9cd5ec875	2011.06.02 12:30:50 (GMT)
ef6751567cbf7c92cd3880fc7aa425c9	2011.07.15 12:30:29 (GMT)
56c06123e34dcc8a8e464da9acd852bb	2011.05.23 11:33:26 (GMT)
a6d549d7c90c412a20fc9e7abc829eb5	2011.07.15 12:30:29 (GMT)
be6f3c214d2a579728fc3537c6454f8c	2011.07.15 12:30:29 (GMT)
0883d6533aa4fb0e40a6e48a66ea84d4	2011.05.23 11:33:26 (GMT)
c3e70e9b50cd3f6cfcd0ac75a60b3464	2011.06.02 12:30:50 (GMT)
75b824c5a6a9b950ccbdaee577fe964b	2011.07.29 10:59:44 (GMT)
9bb26fb5179db8515cdc81cb9f40387d	2011.07.29 10:59:44 (GMT)
d9851c67bfeec5cc37db99be07061857	2011.06.02 12:30:50 (GMT)
07999110cab8c6558be11684d2c02793	2011.07.15 12:30:29 (GMT)
9d5bb8f9441d31148bf4f190e27764cc	2011.06.02 12:30:50 (GMT)
ecd7bec9522e64df7b179b512e71c154	2011.06.02 12:30:50 (GMT)
5e215b9272e4a0ff0d9725ac94bd1541	2011.06.02 12:30:50 (GMT)
9a9dbd2a398fda91167169b0866047d1	2011.07.15 12:30:29 (GMT)

4355f29680630980cf732e46306a39ce	2011.07.15 12:30:29 (GMT)
d4d959bffa33b0e3076421a02e69f13b	2011.07.15 12:30:29 (GMT)
f2bb34acdebcbbd335e6cc2816a0c5f0	2011.06.02 12:30:50 (GMT)
ca25ca44ef0106c4080415f1c2090400	2011.07.15 12:30:29 (GMT)
83ee5deb488d58d924134781e76c416c	2011.06.02 12:30:50 (GMT)
9aa8f3ed12ef1003d24c771af69879f8	2011.07.15 12:30:29 (GMT)
19cc111e41d804f20e5f65c6d0a48953	2011.07.15 12:30:29 (GMT)
acfc7040304b19422ba0a1278b4d9c48	2011.05.23 11:33:26 (GMT)
a515279eee527f7d20f82ef673308151	2011.06.02 12:30:50 (GMT)

Summary

This module is a standalone Windows executable which is automatically started on system boot and runs secretly in the background. It expects removable media arrival and recovers deleted files of attackers' interest from it. This lets the attacker copy deleted docs as well as ZIP and RAR archives from attached removable media.

Initialization

Upon start the module creates a general purpose log file in %TMP%\smrdprev..tmp, where DWORD is a current time value xored with 0x1F3E231, DWORD2 is value of GetTickCount() API call.

So, basically log file path is %TMP%\smrdprev..tmp. Same xor value is used among all variants we have seen (20+ files).

General log starts with internal module name which is essentially "USB_RESTOREv1". It also logs software version which is "0.5" for current module. Other modules had different hardcoded versions. The variants we have seen had **0.3, 0.5** and **0.6**. However, there are few files which didn't have any indication of internal module version and are probably the earliest versions. The log file also includes detailed information about the system it was launched on. Example of system info:

TIME: "2012-11-12 18:07:26 514"
ADMIN: "1"
UserName: "user"
ComputerName: "WIN_XP_105"
TimeZone: "Russian Standard Time"
LOCALE: "Russian_Russia.1251"
OEMCP: "866"
ACP: "1251"
OwnFile: "C:\wusb.exe"

The module changes registry to run automatically during system start:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ctfmon32rt =

If it didn't succeed with HKLM (HKLM) it will try to set similar value in HKCU (HKCU).

Next step, it checks if current user has local administrator rights and logs this as well ("—-ADMIN MODE—-" or "—-USER MODE—-").

One more string is added to the log which indicates end of initialization stage and start of main functionality: "WUSB: begin". This is probably a reference to another module internal name: "WUSB".

Main Procedure

It creates an invisible window named "**sbw**" (Unicode) with window class "**Win32UserHost**". The module calls RegisterDeviceNotification() API to receive systen notifications from particular device GUIDs:

F18A0E88-C30C-11D0-881500A0C906BED8 (GUID_DEVINTERFACE_USB_HUB)
A5DCBF10-6530-11D2-901F00C04FB951ED (GUID_DEVINTERFACE_USB_DEVICE)

So far, the module waits for notifications from USB hubs or USB devices attached to the local system. According to the notification handler code, the module is only interested in new attached volumes except those which were mapped to letter A: (reserved for Floppy drives). Once the new volume appears in the system the module starts a separate thread to process this event. Device is never processed twice until it is plugged in again, the module stores currently mounted devices and handles device removal events properly.

When new USB drive is attached, the module obtains USB device software and vendor ID, drive character, volume name, filesystem type, volume serial number, number of free and used bytes. Then it reads the filesystem using direct disk access and own parser of FAT-based filesystems. It is unable to read other filesystems including NTFS.

Own filesystem parser code allows to avoid using system API to access files, and thus bypass various IDS/IPS software and at the same time look into slack space of the disk. This lets the attackers reveal already deleted files on removable drives. The module creates additional debug log which traces execution events and records all problems that have occured. The filesystem parser log files are located in %TMP%\hsperfdata32sys\bcmntc_rt_*

The parser log includes all discovered filenames, including deleted files. The same module is capable of recovering deleted files. It can recognize file types and recover original files depending on format. It is designed to recognize the following file headers: **DOCX, XLSX, DOC, XLS, RAR, ZIP** . Recovered files are saved in %TMP%\hdbrt32sys\ms32jxtr.dat.

All files created by the module, including logs are stored in custom binary format, which may use Zlib compression, and custom encryption.

This module doesn't communicate with any C&c server, all files are stored locally.

USBStealer module

Known files

MD5	Compilation date (encrypted)
51d5f5a5c7de6a175e269236c2c574b0	2010.10.14 07:07:58 (GMT)
bbe23b8baec0afbd54154820f4a9d7ea	2010.10.14 07:07:58 (GMT)
6abd3d906ebd0e6bf4fb8d00273fdc66	2010.10.14 07:07:58 (GMT)
b9114882ed3a184f8a58284f3fe57cb0	2011.03.02 09:54:14 (GMT)
657f0f4f6183cd2e87fdfd8a88f927c9	2010.10.14 07:07:58 (GMT)
900ab792a9dc9ae35c821cce98164d81	2010.10.14 07:07:58 (GMT)
18bd71030b18f3bc93d08b650ae0d43d	2011.03.02 09:54:14 (GMT)
187adc0380142c61224c53eac9a70955	2011.03.02 09:54:14 (GMT)
78f2c84fefe80bc84361c40d2bbd0501	2010.10.14 07:07:58 (GMT)
b2c60688dc2de4dd4de1f393ae59e317	2010.10.14 07:07:58 (GMT)
3b4125c8dc55ae54fa244a8fdcea8bc9	2011.03.02 09:54:14 (GMT)
760333093fbcc38f6b8d7e1667d192b8	2010.10.14 07:07:58 (GMT)
ffd4096c5d2a2a4801ac6e8ab250a0d0	2011.03.02 09:54:14 (GMT)
92b6b580f1d2e5409a6feb5c8883de2b	2011.03.02 09:54:14 (GMT)
daf244aacbac081693b914a4a1486fa5	2010.10.14 07:07:58 (GMT)
2b08ae138fd27ba62b7ea1e35d38b56f	2010.10.14 07:07:58 (GMT)
48c4e2386cbae6a71b4eccab21ead6e5	2011.03.02 09:54:14 (GMT)
a39636c2fb253ae9ff7b7c0294abf8ac	2011.03.02 09:54:14 (GMT)
f27870dd7bfa952636850a76205f4ba3	2010.10.14 07:07:58 (GMT)
c64343fad7c1f98a8342bd29829fcdf1	2010.06.01 12:13:42 (GMT)
58fbcf7d9146eba51c22e91bdf7128d0	2010.10.14 07:07:58 (GMT)
5c563e849ec86a542daf492b31dde2bb	2010.10.14 07:07:58 (GMT)
4c205fc9c7dbd95316f9ed5aafa34712	2010.10.14 07:07:58 (GMT)
b0e2f3c972477e750d5adbed3650ae81	2010.10.14 07:07:58 (GMT)

Summary

This is a standalone EXE application module which is used to automatically track and steal interesting files on removable disks attached to infected system. It starts automatically on system boot. Criterias for files are defined in external encrypted binary config file which must be deployed during setup. It creates own database of known files' hashes and registers a listener for filesystem changes to do that efficiently.

Initialization

At the start it creates system mutex named ‘ Win32Wbem32Prefetch‘, a system event named “ScxinWordSid_0129211FA” (used to signal end of execution) and a log file at %TMP%\imapispool..0x.ids.

It collects basic system information such as current computer name, current username, and path to original executable module where it started from.

This information is put in the log file in the first place along with current date and time. Every time module adds anything to the log file, it checks if the log file exceeds **15MB size**. If that’s true it deletes current log file and opens a new one using the same path.

It changes registry settings to start automatically on system boot. The changes are made to HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\winns32comp value.

Main Procedure

It creates an invisible window named “sbw” with window class “Win32UserHost“. This window receives notifications from specific device types. The module calls RegisterDeviceNotification() API to listen for notification from the system on particular device GUIDs:

F18A0E88-C30C-11D0-881500A0C906BED8 **(GUID_DEVINTERFACE_USB_HUB)**
A5DCBF10-6530-11D2-901F00C04FB951ED **(GUID_DEVINTERFACE_USB_DEVICE)**

At the same time it create configs monitoring thread which reacts on file changes and reloads new values. The config files are stored in the same directory where current executable module is and named “imapisync32.dat“, “wmilibrt.dat“. imapisync32.dat is encrypted using some custom algorithm, and has information about file targets to track and copy if found. There are common criterias of file size, file age, etc. The module utilizes PCRE (Perl Compatible Regular Expressions) library to effectively apply regular expression white and blacklists against filenames. The criterias are splitted into groups and parameters are names with specific prefix of the group (a, d, s, sd, fa). If group is ended with “a”, that is used as a whitelist, filenames that match against such regexps are to be stolen. Groups ending with “d” represent blacklist, files will be ignored. Below is full decrypted config (imapisync32.dat) with our comments after “#” character:

```
magic=/lddata/ # magic string identifying config file start
_st_size=/300000000/ # max size of the storage (300MB)
_id=/00020/ # internal storage id
_fdeltatime= /2592000/ # files must be not older than 2592000 seconds or 30 days
_max_size=/20000000/ # maximum size of a single file must be not larger than 20MB
_min_size=/1/ # files must not be smaller than 1 byte
a=/\.txt$/
a=/\.csv$/ # Comma-Separated Values, tabular data file
a=/\.eml$/ # exported e-mail file
```

```
a=/\.doc$/
a=/\.vsd$/ # MS Visio document file
a=/\.sxw$/ # StarOffice/OpenOffice document file
a=/\.odt$/ # OpenOffice document file
a=/\.docx$/
a=/\.rtf$/
a=/\.pdf$/
a=/\.mdb$/ # MS Access database file
a=/\.doc\./ # *.doc.* files, could be for an archive of MS Word document file
a=/\.odt\./ # *.odt.* files, could be for an archive of OpenOffice document file
a=/\.docx\./ # etc
a=/\.rtf\./
a=/\.pdf\./
a=/\.xls\./
a=/\.wab$/ # Windows Address Book file (used in Outlook Express)
a=/accounts\.ini/ # could be Opera browser accounts and settings file
a=/account\.cfn/ # TheBat! e-mail client accounts and settings file
a=/signons\.txt/ # saved user names and passwords in Thunderbird, Sunbird, and
earlier versions of Firefox
a=/ScribeOptions\.xml/ # probably settings of Scribe crossplatform e-mail client
a=/wand\.dat/ # Opera browser password manager database
a=/bpftp\.dat/ # BulletProof FTP client password database
a=/sm\.dat/ # CuteFTP password database
a=/smdata\.dat/ # CuteFTP password database
a=/FileZilla\.xml/ # FileZilla FTP client password and settings database
a=/ftplist\.txt/ # TotalCommander ftp upload file list (may contain credential
information)
a=/addrbk\.dat/ # TurboFTP password and settings file
a=/wcx_ftp\.ini/ # Total Commander cached FTP credentials database
a=/ws_ftp\.ini/ # WS_FTP client password and settings file
a=/andrq\.ini/ # &RQ ICQ client password and settings file
a=/account\.xml/ # Very generic name, used in various software
a=/odigo\.com\.odu/ # Odigo instant messenger settings file
a=/TheBee\.ini/ # Some “The Bee” software ini file, unclear which software it is

# The following subgroup defines useless files that will not be taken.

d=/\\~wordspool.*\.tmp\.doc$/
d=/\\~wordspool.*\.srt\.doc$/
d=/\\~wordspool.*\.rtc\.doc$/

_s_fctime= /1990-01-01 01:02:03/ # files of group “s” after 1990-01-01 are interesting
_s_max_size=/20000000/ # max size for group “s” is 20MB as well
_s_min_size=/1/ # min size is 1 byte

# Some patterns of filenames below seem to be related to some other malware seen
on usb drives. It may contain stolen credentials, so they copy it as well.
```

sa=/.*mssysmgr\.ocx/
sa=/.*\.cab\.bak/
sa=/.*list\.tlb/
sa=/.*drive\.tlb/
sa=/.+\\\$lddata\$\\.+/
sa=/.+\\NT.Config\\.+/
sa=/.*\\ldupver\.txt/
sa=/\w:\\[\d\w]+\.dll/
sa=/\w:\\[\d\w]+\.exe/
sa=/.*autorun\.inf/
sa=/.*thumb\.dd/
sa=/.*thumb\.db/
sa=/.*msnmsngr\.exe/
sa=/.*svchost\.exe/
sa=/.*EXPLORER\.EXE/
sa=/.+\.iau/
sa=/.+\.rst/
sa=/\.xps/ # This is a subgroup of various files with secrets, such as digital certificates, configs and password databases
sa=/\.cif$/
sa=/\.key$/
sa=/\.crt$/
sa=/\.cer$/
sa=/\.hse$/
sa=/\.pgp$/
sa=/\.gpg$/
sa=/\.conf$/
sa=/passw/
sa=/secret/
sa=/crypt/
sa=/krypt/
sa=/cypher/
sa=/cipher/
sa=/\.ovpn$/
sa=/\.xia$/
sa=/\.xiu$/
sa=/\.xis$/
sa=/\.xio$/
sa=/\.xig$/
sa=/pubring.*/
sa=/secring.*/
sa=/\.acidcsa$/
sa=/\.acidsca$/
sa=/\.aciddsk$/
sa=/\.acidpvr$/
sa=/\.acidppr$/

```
sa=/\.acidssa$/
sa=/\\ACIDInstallv.*\.exe$/
sa=/\\ACIDdirInstallv.*\.exe$/ sa=/\\Acid Technologies\\/ # Probably this and above refers to Acid Cryptofiler military grade encryption software
sa=/.*cisco.*\.txt$/
sa=/.*cisco.*\.cfg$/
_f_fctime= /1999-02-03 02:03:07/
_f_max_size=/10000000/
_f_min_size=/1/
fa=/~WRD32cache.*\.doc$/
fa=/glxcrt.*\.dat/
fa=/bcmntc.*\.tmp/
fa=/smrdprev.*\.tmp/
a=/\.ttr$/ # maybe TTR Backup software?
a=/\.tbe$/ # unclear
a=/\.tte$/ # unclear
a=/\._ok$/ # unclear
a=/\.ki$/ # unclear
a=/\.txt$/
a=/\.tvt$/
a=/\.txr$/ # CorelDraw file?
a=/\.tvr$/
a=/\.tbr$/
a=/\.kin$/
```

Config file may change during module work and there is a special thread that monitors config file changes and loads new configuration when it is changed.

The module lists all available attached removable drives, iterates through all of them except A: and B: (reserved for floppy disks). It collects full file paths and file meta-information, compresses, encrypts and stores this in %TMP%\mrtdsync\win64esrt_* database. Additionally, files from the list of interest are copied to win64berts* files, which also have full information and body of the file. It is encrypted and compressed twice.

There is a separate thread to monitor filesystem changes and every time new file/directory is created or changed on the monitored drives updated information goes to %TMP%\mrtdsync\win64esrt_* in separately spawned thread, additionally all changed or created files are copied to a new win64berts* file.

This module works offline and doesn’t communicate with any other host on the network.

6. Keyboard group

Keylogger module

Known variants:

MD5	Compilation date (encrypted)	Compilation date (payload)
33bda0e77b840809e66e12d020e054c5	2011.04.14 12:00:26 (GMT)	2010.10.05 10:53:49 (GMT)
3cb7318ed40239f7219d86343a17b54b	2011.05.10 07:45:02 (GMT)	2010.10.05 10:53:49 (GMT)
dfcce19f2852db652071088bf9461b4a	2011.05.10 07:44:55 (GMT)	2010.10.05 10:53:49 (GMT)
6079a0746e76c1090dc110e08de645e2	2011.05.20 11:00:18 (GMT)	2010.10.05 10:53:49 (GMT)
57897c997c699135b9460c0be7a4b27e	2011.10.10 07:59:41 (GMT)	2010.10.05 10:53:49 (GMT)
ecc7a5ef3f5e92f0c7da0bef8d392b5f	2011.05.12 10:47:39 (GMT)	2010.10.05 10:53:49 (GMT)

The file is a PE EXE file, compiled with Microsoft Visual Studio 2005. Its main purpose is to log keystrokes, copy input texts and make screenshots.

Main function

Opens and Creates Event WIN_%08X%08X%08X%08X%08X (SHA1(first 512 bytes of self file)), if exists, terminates. Sets in AutoRun, using the following registry keys:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
“Mspmr32” = %Path to the module%

Deletes file “Keylogger.log” (this filename is not used anywhere else). Registers a window class with RegisterClassEx API using ClassName “svchost.exe“, assigns a window procedure that implements main module functionality. Creates window with CreatWindowEx API using ClassName “svchost.exe“, window name “svchost” and associates it with a registered class.

Registers a device with RegisterRawInputDevices API and associates it with created window with a flag RIDEV_INPUTSINK which enables the caller to receive the input even when the caller is not in the foreground.

Window procedure

All the logging actions are implemented basing on receiving WM_INPUT window message, if raw input for GetRawInputData API comes from keyboard and GetRawInputData received WM_KEYDOWN message.

Collected information

It collects some general information about current user and opened windows/processes:

Foreground window text or WT_UNKNOWN
Module FileName or MN_UNKNOWN
Foreground window class name

UserName

It is capable of making full desktop screenshots, copying clipboard data of password input fields (to check a window EM_GETPASSWORDCHAR is sent to the window).

If “Shift+Insert”, “Ctrl+C”, “”Ctrl+V”,”Ctrl+X”, “Ctrl+Insert” is pressed then it copies clipboard data.

Creates File %TMP%\SSDPserv32\ssdtrbs%08x%.sys.%d% (%08x – Random Hex value, %d – time64()). All the collected information is compressed with Zlib and RC4 encrypted with the key “qefwljkfnw3l;fjwe;fklwejfw;eflkwe;flwe”and written to this file.