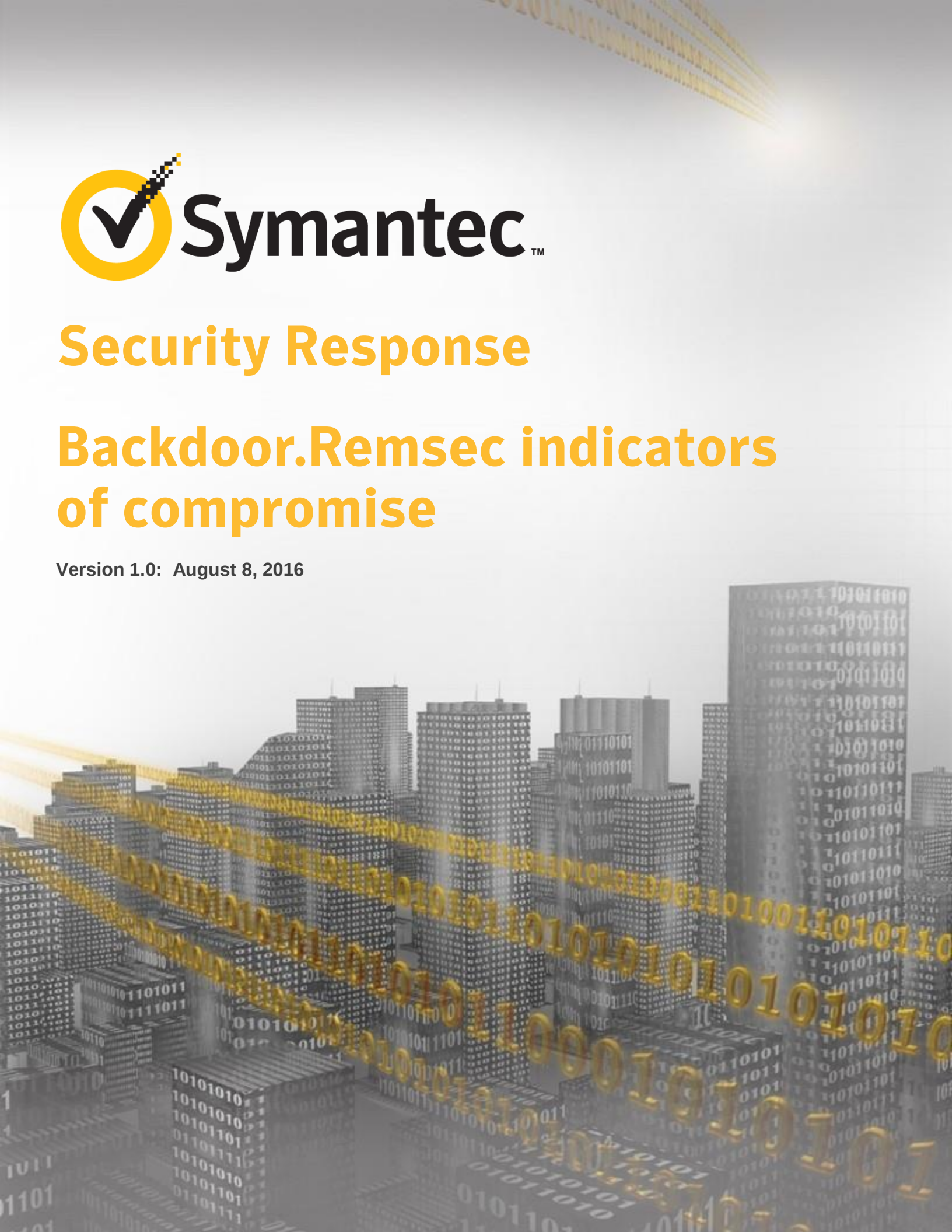

Security Response

Backdoor.Remsec indicators of compromise

Version 1.0: August 8, 2016

Backdoor.Remsec

The Backdoor.Remsec signature is used to detect several different components. These various components work together as a framework to provide an attacker complete control over a victim computer, allow them to spread across a network, utilize a discreet command and control protocol, and deploy custom tools as required.

Only a subset of Backdoor.Remsec components have been discovered and documented. Some samples have not been fully analyzed. Functionality is mostly implemented in the form of modules downloaded over a network connection, and then executed in memory, and many of these components have not been retrieved.

Several different components have been discovered on victim computers. Many of these components share functionality, code, or other elements that, as well as being present on victims' computers, links them.

- **Loader**: Seen with the filename MSAOSSPC.DLL, this component is responsible for loading files from disk and executing them. The files on disk contain the payload in a specific format we are calling the executable blob. The component also logs data. Executable blobs and data are encrypted and decrypted with a repeating key of 0xBAADF00D.
- **Lua modules**: Several components use a Lua interpreter with Lua scripts to implement their functionality. The Lua components are stored in the same executable blob format that the loader works with. Several different Lua modules have been retrieved and their functionality includes:
  - Network loader - This loads an executable over the network for execution. It may use RSA and RC6.
  - Host loader - Loads at least three components, kblog, ilpsend, and updater.
  - Keylogger - Exfiltrates keylog data. It also contains the string SAURON, which may be a code word to describe the module or project.
- **Network listener:** A number of samples that implement different techniques for opening a network connection based on monitoring for specific types of traffic. This includes ICMP, PCAP, and RAW network sockets. It's unclear exactly what is checked for with PCAP and Raw, however, the ICMP listener checks for echo and echo response packets.
- **Named pipe back door**: A minimal back door controlled over named pipes. This can execute data in the format of the executable blob, or standard PE files.
- **A second named pipe back door** - This offers several more commands than the other named pipe back door, including sending the executable blob, listing files, and reading/writing/deleting files.
- **HTTP back door**: Includes several URLs for C&C servers.

Loader

This description is based on the analysis of the file with the MD5 hash of 2a8785bf45f4f03c10cd929bb0685c2d which was seen with the file name of MSAOSSPC.DLL.

As mentioned, the loader is responsible for loading files from disk and executing them.

The loaded files adhere to a specific format referred to as the executable blob. That format is shown in Table 1.

Offset	Size	Purpose
0	vary	*Executable_Blob_Header*, described below
vary	variable	blob of executable code

Table 1. Loaded files format

The *Executable_Blob_Header* can be interpreted as the following structure:

Offset	Size	Purpose
0	Dword	Magic, enforcing value C102AA02
4	Byte	Version_Major (guess), enforcing value 2
5	Byte	Version_Minor (guess), we observed values 0, 1 and 3 in analyses samples
6	Dword	Entry_32, entry point for 32-bit mode represented as offset from start of Executable_Blob structure
0A	Dword	Entry_64, entry point for 64-bit mode represented as offset from start of Executable_Blob structure
0E	Dword	Tag (guess), introduced starting with Version_Minor 1, we observed values 0 and 7B3924B1 in analyzed samples
12	Byte	unknown purpose, introduced after Version_Minor 1, but not later than Version_Minor 3, observed only value 0 in analysed samples

Table 2. Executable_Blob_Header structure

The loaded executable blobs may be retrieved from the following path:

- c:\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\RP0\A0000002.dll

Persistence

The loader is implemented as a (fake) Security Support Provider. Implementing the export of *InitSecurityInterfaceW* effectively functions as a loadpoint for the module.

Injection library

This loader contains a relatively advanced library supporting injections into both 32-bit and 64-bit targets. The implementation includes multiple redundant capabilities, including an embedded implementation of *CreateRemoteThread* with *CSR* notification by way of *CSRSS_CREATE_THREAD_LPCMESSAGE*.

Payload structure

The payload file, referred to by the attackers as a "module" per error messages, contains an encrypted structure storing the following:

- *PE_Executable* with only minimal verification by checking *Machine* field from *IMAGE_FILE_HEADER*
- *Helper_Blob,* described below

Based on checks implemented by the *Loader*, the *Helper_Blob* can be interpreted as the following structure of length *Helper_Blob_Size*:

Offset	Size	Purpose
0	Dword	*Data_Blob_Size*, enforcing *Data_Blob_Size* < *Helper_Blob_Size* - 4
4	*Data_Blob_Size* Bytes	*Data_Blob*
variable	variable	*PE_Loader* responsible for loading *PE_Executable*

Table 3. Helper_Blob structure

The *PE_Executable* and *Helper_Blob* are serialized as the following variable-length structure:

Offset	Size	Purpose
0	Dword	*Version*, referred as "module file version" in error messages, enforcing value 1
4	Dword	*PE_Executable_Size*
8	Dword	*Helper_Blob_Size*
0C	*PE_Executable_Size* Bytes	*PE_Executable*
variable	*Helper_Blob_Size* Bytes	*Helper_Blob* described above

Table 4. Payload_Blob structure

The *Payload_Blob* structure is compressed using a *ZLIB* algorithm.

The compressed blob is then encrypted using an *RC5* algorithm in *CBC* mode with the following parameters:

- word size: 32
- rounds: 12
- key: "0xBAADF00DBAADF00DBAADF00DBAADF00D"
- iv: zero bytes

Notice that encryption may require padding for a compressed blob. The encrypted blob is stored on disk as a payload file.

Log structure

The log file contains messages generated by the *Loader* stored in encrypted form.

The *Log_Messages* are represented as Unicode text, for example:

```
Starting log (computer: "XXX" serial: 80B0:51E5 timezone: UTC+0h).
Failed to open module c:\System Volume Information\_restore{ED650925-A32C-
4E9C-8A73-8E6F0509309A}\RP0\A0000002.dll: (2) The system cannot find the
file specified.
Ending log.
```

The *Log_Messages* are compressed using a *ZLIB* algorithm.

The compressed log is then encrypted using an *RC5* algorithm in *CBC* mode with the following parameters:

- word size: 32
- rounds: 12
- key: "0xBAADF00DBAADF00DBAADF00DBAADF00D"
- iv: zero bytes

Lua modules

A number of samples were discovered that contained a modified version of a Lua interpreter, and compiled Lua scripts that would be executed by the interpreter. Each sample has the Lua script and plugins stored in an encrypted configuration blob.

These configuration blobs are encrypted using one of two distinct methods. These two techniques were labelled version A and version B (vA, and vB for short).

The pre-compiled Lua scripts use customized binary format:

- Lua_SIGNATURE was modified to avoid "Lua" magic value
- Representation of char variables was modified, possibly to simplify handling of Unicode characters

A number of samples that implement various different functionalities were discovered. These are described below:

Network loader

The functionality of the component with the MD5 hash of 90b4b5f0a475f3a028be2f71409e6d1a is to receive executable modules from remote attackers and run them from memory on the local computer.

This component can use one of the following methods to establish a communication channel (depending on passed parameters):

- Listen for an incoming TCP connection on an arbitrary port
- Connect to an arbitrary host over TCP
- Communicate over a handle provided by the caller

The communication is encrypted using *RSA* and *RC6*.

The received executable module follows the PE format and provides the following exports:

- *init*, where two constant values, 2 and 1, are passed
- *main*, where a specific structure is passed, including functions to communicate over an established encrypted communication channel

The following is the Lua script referring to this component:

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

Host loader

Two version of the host loader component were discovered. These files had the MD5 hashes of 7261230a43a40bb29227a169c2c8e1be and 48d0c8faaee08fc51346925090af89aa.

This component contains configuration data. The data for both files discovered is shown in Tables 5 and 6.

DLL_NAME	nseci.dll
BASE_STORAGE	c:\\System Volume Information\\{aa112c99-f343-4107-8ba1-22951714a641}
BLOB_STORAGE	c:\\System Volume Information\\{951841cb-d1a4-4d7c-b44e-2c3d25996e37}
KBLOG_UUID	{85f24f97-7321-4849-8c78-5989f5837ad4}
ILPSEND_UUID	{4d19edb2-5391-4f14-b27f-a3d553f411f4}
UPDATER_UUID	{6d46df72-115e-4c4c-a0a5-510dfe46f8aa}
TMP_STORAGE	c:\\System Volume Information\\{b68475dd-ed80-4cc4-b508-77314abfafa0}
SPOOL_STORAGE	c:\\System Volume Information\\{e69c37ac-a8ac-4827-8ce8-3126748e23bb}
STATE_FILE	c:\\System Volume Information\\{c089b325-4a8b-498b-bc12-51d325b91387}
BUS_LOG_STORAGE	c:\\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\\RP0
BUS_STORAGE	c:\\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\\RP1

Table 5. Configuration data for host loader (version with MD5: 7261230a43a40bb29227a169c2c8e1be)

DLL_NAME	dsecsp.dll
MUTEX_NAME	Global\\{b3898039-f3d8-4965-b618-a8a0d031cc5a}
BASE_STORAGE	c:\\System Volume Information\\{9663c974-3112-4367-9c2a-06afbb7a67ce}
BLOB_STORAGE	c:\\System Volume Information\\{1864d9b7-0068-4979-87dc-c9668a9a2ae6}

KBLOG_UUID	{85f24f97-7321-4849-8c78-5989f5837ad4}
ILPSEND_UUID	{4d19edb2-5391-4f14-b27f-a3d553f411f4}
UPDATER_UUID	{6d46df72-115e-4c4c-a0a5-510dfe46f8aa}
TMP_STORAGE	c:\\System Volume Information\\{69785f90-a546-4ee0-8f1e-41ada23dbfcb}
SPOOL_STORAGE	c:\\System Volume Information\\{c0f55b36-0a62-400b-acbd-1a9624132f88}
STATE_FILE	c:\\System Volume Information\\{40a58472-645d-48d3-a150-54e049cd8166}
UPDATER_REPLAY_LOG	c:\\System Volume Information\\{54af39be-c3cd-4ee8-b2cb-6bfb3314b5b3}
BUS_LOG_STORAGE	c:\\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\\RP0
BUS_STORAGE	c:\\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\\RP1

Table 6. Configuration data for host loader (version with MD5: 48d0c8faaee08fc51346925090af89aa)

The component acts as a loader for the following sub-modules:

- kblog
- ilpsend
- updater

The above components are loaded from files stored in the path referenced by the BLOB_STORAGE variable in the configuration. These are decrypted and injected into running processes. A kblog component was retrieved during the investigation which, as described below, is a keylogger. The other modules were not retrieved, however it is very likely the ilpsend module is used to exfiltrate the keylogger data over HTTPS or SMTP, and the update component is used to download updated versions of the module.

Each Lua script also includes version information:

- 7261230a43a40bb29227a169c2c8e1be: VERSION = "4.0"
- 48d0c8faaee08fc51346925090af89aa: VERSION = "5.2"

Keylogger

The following description is for the file with the MD5 hash of 6cd8311d11dc973e970237e10ed04ad7.

The keylogger logs data to the following locations:

- "%WINDIR%\\temp\\bka*.da"
- "%WINDIR%\\temp\\bka*.dat"
- "C:\\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\\RP0\\change.log.*"
- "C:\\System Volume Information_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\\RP1\\A*"

The keylogger component may be referred to as SAURON by the attackers (based on SAURON_KBLOG_KEY). Collected data is exfiltrated using the ILPS module.

```
KBLOG_ROTATE_SECS = 10800
tmp_dir = (os.getenv)("WINDIR") .. "\\temp\\"
drive = "C:\\"
SAURON_KBLOG_KEY = "mISfx1q2Ef/QJPO4gi6DMKD51xeQ380knDrULcZyTF5vFNWbUvT23PX9LrI
R7oDjKOcd9Y97XehAkmqnVW4r4Gbsk0hkjjRFZ/I7lO2eK8eE2mcSW+TRBMJBPEJEw=="
create_log = function(dir, key, soft_limit, hard_limit)
```

Figure 1. String referencing Sauron in Remsec keylogger module

Network listeners

A number of network listener components were discovered. These components used varying techniques to listen to network traffic, specifically looking for commands issued by an attacker. The commands are in the form of (encrypted) executable code. The MD5 hashes of the files discovered are listed in Table 7.

MD5
0a0948d871ef5a3006c0ab2997ad330e
113050c3e3140bf631d186d78d4b1dc0
1d9d7d05ab7c68bdc257afb1c086fb88
1f316e14e773ca0f468d0d160b5d0307
7b8a3bf6fd266593db96eddaa3fae6f9
cf6c049bd7cd9e04cc365b73f3f6098e
7c3eecfb5174ca5cb1e03b8bf4b06f19

Table 7. MD5 hashes of network listener compononents discovered

The samples provide a general-purpose framework to execute arbitrary modules sent by the attackers. These modules are apparently in-memory, but some filesystem code is also present. The discovered samples use different combinations of network listeners. Components collected so far support PcapUdp, PcapTcp, Pcap, Icmp, Dns, Raw, Pipe, and Http.

As an example, the ICMP listener opens a RAW socket and listens for ICMP echo request or response packets. If any such packet is discovered, the malware will attempt to decrypt the contents of the ICMP message. If the contents decrypt correctly, the decrypted code is then executed by the malware.

Named pipe

The following description is for the file with the MD5 hash of 9f81f59bc58452127884ce513865ed20.The sample provides a minimal back door controlled over named pipes. The following functionality is available to the remote attacker:

- Send *Executable_Blob* containing arbitrary code to be executed
- Run arbitrary commands using *CreateProcessW* API

Additional named pipe back doors with more functionality have also been discovered. These other back doors are able to process more commands. The extra commands are listed below, per each sample.

Command	Description
1	Read and optionally delete arbitrary file.
2	Write arbitrary file.
3	List content of arbitrary folder.
4	Delete arbitrary file.
5	Contains *Executable_Blob* with arbitrary code to be executed.
6	Exit (guess).

Table 8. Back door commands per sample 7001A747EED1B2DA1C863B75500241F7

Command	Description
1	Read and optionally delete arbitrary file.
2	Write arbitrary file.
3	List content of arbitrary folder.
4	Delete arbitrary file.
5	Contains *Executable_Blob* with arbitrary code to be executed.

6	Sets a flag - purpose unconfirmed.
7	Start TCP proxy by: connecting to arbitrary IPv4 endpoint address or binding new socket to local IPv4 endpoint address selected by the attacker and accepting connection The proxy will forward communication between the *Named Pipe* and the TCP connection.
8	Contains blob with arbitrary code to be executed. This blob does NOT follow the *Executable_Blob* format. Communication between *Back door* and executed blob is apparently by way of created TCP connection.
9	Check computer architecture using *IsWow64Process* API.

Table 9. Back door commands per sample 2cf1f878ab4eb2f043ccae7d653770c8

HTTP back door

A back door that uses HTTP for it's command and control functionality was found on several victims' computers. Two variants were discovered, files the with MD5 hashes of edb9e045b8dc7bb0b549bdf28e55f3b5 and 01ac1cd4064b44cdfa24bf4eb40290e7.

The configuration information in those samples included the following URLs:

edb9e045b8dc7bb0b549bdf28e55f3b5:

- http[://]flowershop22.110mb.com/shop.php
- http[://]wildhorses.awardspace.info/hindex.php
- http[://]www.myhomemusic.com/music.php

01ac1cd4064b44cdfa24bf4eb40290e7:

- http[://]www.myhomemusic.com/mymusic.php
- http[://]flowershop22.110mb.com/flowers.php
- http[://]wildhorses.awardspace.info/horses.php

Indicators of compromise

Backdoor.Remsec hashes

MD5	SHA256
2a8785bf45f4f03c10cd929bb0685c2d	6c8c93069831a1b60279d2b316fd36bffa0d4c407068dbef81b8e2fe8fd8e8cd
171f39bd2f79963b5ec2b588b42da034	d629aa328fef1bd3c390751575f65d2f568b4b512132d77ab3693709ae2d5c84
44879e5240fbe41c909c59abdcc678bc	9035a1e71c87620ead00d47c9db3768b52197703f124f097fa38dd6bf8e2edc8
bf208df25db6ef67639765b2f0fc2c8c	36b74acba714429b07ab2205ee9fc13540768d7d8d9d5b2c9553c44ea0b8854f
beb2cc1694d89354a062b04b27811099	0f8af75782bb7cf0d2e9a78af121417ad3c0c62d8b86c8d2566cdb0f23e15cea
113050c3e3140bf631d186d78d4b1dc0	bde264ceb211089f6a9c8cfbaf3974bf3d7bf4843d22186684464152c432f8a5
546a2ebb0100ebff6c150fae49b87187	4a15dfab1d150f2f19740782889a8c144bd935917744f20d16b1600ae5c93d44
7b8a3bf6fd266593db96eddaa3fae6f9	3782b63d7f6f688a5ccb1b72be89a6a98bb722218c9f22402709af97a41973c8
cf6c049bd7cd9e04cc365b73f3f6098e	6b06522f803437d51c15832dbd6b91d8d8b244440b4d2f09bd952f335351b06d
0886ace08961e71e5a572698307efdee	96e6b2cedaf2840b1939a9128751aec0f1ac724df76970bc744e3043281d3afd
7c3eecfb5174ca5cb1e03b8bf4b06f19	02a9b52c88199e5611871d634b6188c35a174944f75f6d8a2110b5b1c5e60a48
0a0948d871ef5a3006c0ab2997ad330e	ab8181ae5cc205f1d3cae00d8b34011e47b735a553bd5a4f079f03052b74a06d
1d9d7d05ab7c68bdc257afb1c086fb88	c8f95bf8a76ff124cc1d7a8439beff360d0eb9c0972d42a8684c3bd4e91c6600
1f316e14e773ca0f468d0d160b5d0307	9572624b6026311a0e122835bcd7200eca396802000d0777dba118afaaf9f2a9

7261230a43a40bb29227a169c2c8e1be	d737644d612e5051f66fb97a34ec592b3508be06e33f743a2fdb31cdf6bd2718
234e22d3b7bba6c0891de0a19b79d7ea	30a824155603c2e9d8bfd3adab8660e826d7e0681e28e46d102706a03e23e3a8
6cd8311d11dc973e970237e10ed04ad7	a4736de88e9208eb81b52f29bab9e7f328b90a86512bd0baadf4c519e948e5ec
01ac1cd4064b44cdfa24bf4eb40290e7	8e63e579dded54f81ec50ef085929069d30a940ea4afd4f3bf77452f0546a3d3
edb9e045b8dc7bb0b549bdf28e55f3b5	96c3404dadee72b1f27f6d4fbd567aac84d1fdf64a5168c7ef2464b6c4b86289
7001a747eed1b2da1c863b75500241f7	04ea378405c9aa879478db3d6488ce79b694393501555ccabc109fa0f4844533
9f81f59bc58452127884ce513865ed20	720195b07c81e95dab4a1469342bc723938733b3846d7647264f6d0816269380
58e770a9630e13129b4187cfcada76d0	2f128fff48d749f08786e618d3a44e2ac8020cc2ece5034cb1079901bbde6b7e
65823a7f4c545cc64d7d478dd6866381	6189b94c9f3982ce15015d68f280f5d7a87074b829edb87825cadab6ec1c7ec2

Table 9. Hashes associated with Backdoor.Remsec

Network

- http[://]flowershop22.110mb.com/flowers.php
- http[://]flowershop22.110mb.com/shop.php
- http[://]wildhorses.awardspace.info/hindex.php
- http[://]wildhorses.awardspace.info/horses.php
- http[://]www.myhomemusic.com/music.php
- http[://]www.myhomemusic.com/mymusic.php

Developing a network signature for the ICMP component is not straightforward, however it may be possible to identify suspect ICMP packets as follows:

- Packet size greater than 0x1c
- ICMP Echo or Response
- High entropy payload, at least non-numbers or letters. (Standard ICMP packets will typically contains ASCII characters)

Backdoor.Remsec Yara signatures

```
rule remsec_executable_blob_32
{
    meta:
        copyright = "Symantec"
    strings:
        $code =
        /*
            31 06                       l0: xor     [esi], eax
            83 C6 04                        add     esi, 4
            D1 E8                           shr     eax, 1
            73 05                           jnb     short l1
            35 01 00 00 D0                  xor     eax, 0D0000001h
            E2 F0                       l1: loop    l0
        */
        {
            31 06
            83 C6 04
            D1 E8
            73 05
            35 01 00 00 D0
```

```
            E2 F0
        }
    condition:
        all of them
}
rule remsec_executable_blob_64
{
    meta:
        copyright = "Symantec"
    strings:
        $code =
        /*
            31 06                       l0: xor     [rsi], eax
            48 83 C6 04                     add     rsi, 4
            D1 E8                           shr     eax, 1
            73 05                           jnb     short l1
            35 01 00 00 D0                  xor     eax, 0D0000001h
            E2 EF                       l1: loop    l0
        */
        {
            31 06
            48 83 C6 04
            D1 E8
            73 05
            35 01 00 00 D0
            E2 EF
        }
    condition:
        all of them
}
rule remsec_executable_blob_parser
{
    meta:
        copyright = "Symantec"

    strings:
        $code =
        /*
            0F 82 ?? ?? 00 00               jb      l_0
            80 7? 04 02                     cmp     byte ptr [r0+4], 2
            0F 85 ?? ?? 00 00               jnz     l_0
            81 3? 02 AA 02 C1               cmp     dword ptr [r0],
0C102AA02h
            0F 85 ?? ?? 00 00               jnz     l_0
            8B ?? 06                        mov     r1, [r0+6]
        */
        {
          ( 0F 82 ?? ?? 00 00 | 72 ?? )
            ( 80 | 41 80 ) ( 7? | 7C 24 ) 04 02
          ( 0F 85 ?? ?? 00 00 | 75 ?? )
            ( 81 | 41 81 ) ( 3? | 3C 24 | 7D 00 ) 02 AA 02 C1
          ( 0F 85 ?? ?? 00 00 | 75 ?? )
            ( 8B | 41 8B | 44 8B | 45 8B ) ( 4? | 5? | 6? | 7? | ?4 24 |
?C 24 ) 06
        }
```

```
    condition:
        all of them
}
rule remsec_encrypted_api
{
    meta:
        copyright = "Symantec"

    strings:
        $open_process =
        /*
            "OpenProcess\x00" in encrypted form
        */
        { 91 9A 8F B0 9C 90 8D AF 8C 8C 9A FF }
    condition:
        all of them
}
rule remsec_packer_A
{
    meta:
        copyright = "Symantec"

    strings:
        $code =
        /*
            69 ?? AB 00 00 00               imul    r0, 0ABh
            81 C? CD 2B 00 00               add     r0, 2BCDh
            F7 E?                           mul     r0
            C1 E? 0D                        shr     r1, 0Dh
            69 ?? 85 CF 00 00               imul    r1, 0CF85h
            2B                              sub     r0, r1
        */
        {
            69 ( C? | D? | E? | F? ) AB 00 00 00
            ( 81 | 41 81 ) C? CD 2B 00 00
            ( F7 | 41 F7 ) E?
            ( C1 | 41 C1 ) E? 0D
            ( 69 | 45 69 ) ( C? | D? | E? | F? ) 85 CF 00 00
          ( 29 | 41 29 | 44 29 | 45 29 | 2B | 41 2B | 44 2B | 45 2B )
        }
    condition:
        all of them
}
rule remsec_packer_B
{
    meta:
        copyright = "Symantec"

    strings:
        $code =
        /*
            48 8B 05 C4 2D 01 00            mov     rax, cs:LoadLibraryA
            48 89 44 24 48                  mov     qword ptr
[rsp+1B8h+descriptor+18h], rax
            48 8B 05 A0 2D 01 00            mov     rax, cs:GetProcAddress
```

```
            48 8D 4C 24 30                  lea     rcx,
[rsp+1B8h+descriptor]
            48 89 44 24 50                  mov     qword ptr
[rsp+1B8h+descriptor+20h], rax
            48 8D 84 24 80 00 00 00         lea     rax,
[rsp+1B8h+var_138]
            C6 44 24 30 00                  mov     [rsp+1B8h+descriptor],
0
            48 89 44 24 60                  mov     qword ptr
[rsp+1B8h+descriptor+30h], rax
            48 8D 84 24 80 00 00 00         lea     rax,
[rsp+1B8h+var_138]
            C7 44 24 34 03 00 00 00         mov     dword ptr
[rsp+1B8h+descriptor+4], 3
            2B F8                           sub     edi, eax
            48 89 5C 24 38                  mov     qword ptr
[rsp+1B8h+descriptor+8], rbx
            44 89 6C 24 40                  mov     dword ptr
[rsp+1B8h+descriptor+10h], r13d
            83 C7 08                        add     edi, 8
            89 7C 24 68                     mov     dword ptr
[rsp+1B8h+descriptor+38h], edi
            FF D5                           call    rbp
            05 00 00 00 3A                  add     eax, 3A000000h
        */
        {
            48 8B 05 ?? ?? ?? ??
            48 89 44 24 ??
            48 8B 05 ?? ?? ?? ??
            48 8D 4C 24 ??
            48 89 44 24 ??
            48 8D ( 45 ?? | 84 24 ?? ?? 00 00 )
          ( 44 88 6? 24 ?? | C6 44 24 ?? 00 )
            48 89 44 24 ??
            48 8D ( 45 ?? | 84 24 ?? ?? 00 00 )
            C7 44 24 ?? 0? 00 00 00
            2B ?8
            48 89 ?C 24 ??
            44 89 6? 24 ??
            83 C? 08
            89 ?C 24 ??
          ( FF | 41 FF ) D?
          ( 05 | 8D 88 ) 00 00 00 3A
        }
    condition:
        all of them
}
```

About Symantec

Symantec Corporation is the global leader in cybersecurity. Operating one of the world's largest cyber intelligence networks, we see more threats, and protect more customers from the next generation of attacks. We help companies, governments and individuals secure their most important data wherever it lives.

For specific country offices and contact numbers, please visit our website.

Symantec Corporation World Headquarters
350 Ellis St.
Mountain View, CA 94043 USA
+1 (650) 527-8000
1 (800) 721-3934
www.symantec.com