Carbon Black Linux Sensor Troubleshooting Guide

1 Overview

The Carbon Black Linux Sensor gathers operating-system level activity in the form of process creations and terminations, file modifications, inbound and outbound network communications, and module (code) loads. Additionally, metadata about an executable module is gathered, as well as a copy of each unique executable module.

Core activity collection is accomplished via proprietary kernel modules. All administrative overhead, including communications with the Carbon Black server, is accomplished via a usermode service.

All gathered data is communicated with the Carbon Black Enterprise Server for storage, indexing, and analysis.

2 Supported Linux Versions

OS Version	Architecture	Additional Notes
CentOS/Red Hat Enterprise Linux 6.4	x64	Tested mostly with kernel 2.6.32-358.18.1.el6.x86_64. Some testing with 2.6.32-358.el6.x86_64
CentOS/Red Hat Enterprise Linux 6.5	x64	Tested mostly with the most recent kernels above 2.6.32-431.17.1.el6 and higher
CentOS/Red Hat Enterprise Linux 6.6	x64	Tested with just released 2.6.32-504.el6

3 Carbon Black Linux Sensor Version History

Sensor Version	Release Date	Additional Notes
4.2.1		GA release

4 Installation

Please see the Linux sensor install guide.

5 Uninstallation

Uninstallation can be achieved by running the following command:

```
      /opt/cbsensor/sensoruninstall.sh
```

In the rare case, the sensor can be uninstalled manually, via ‘rpm -e cbsensor’. Note that the sensor data and logs will remain.

6 Troubleshooting

6.1 General Logging

The user mode portion of the sensor creates an execution logs under

```
      /var/log/cb/sensor/cbdaemon.INFO
```

This log file is a symbolic link which is recreated each time the daemon runs. The default log level is set to WARNING. This will result in the generation of log files for WARNING and ERROR levels:

```
      /var/log/cb/sensor/cbdaemon.WARNING
      /var/log/cb/sensor/cbdaemon.ERROR
```

The kernel module logs messages to /var/log/messages. Type the following command in terminal to dump kernel messages in real time:

```
      sudo tail -f /var/log/messages | grep CbSensor
```

6.2 Kernel Panic Recover

If you are experiencing kernel panics on boot, or are unable to uninstall the Carbon Black sensor, it may be advisable to manually disable the Carbon Black sensor:

1. Boot your PC using the Centos Installation CD.
2. At the CD’s menu, select ‘Rescue installed system’
3. Select the defaults for all the options during the rescue wizard prompts. (select

‘Continue’ at the prompt to find and mount your Linux installation)
4. Select the ‘Start a shell’ option
5. At the shell prompt change directory to /mnt/sysimage/etc/sysconfig/modules/
6. Delete or rename the ‘cbsensor.modules’ file
7. Reboot

6.2.2 Kernel Panic Data Collection

If you experience a panic while using the sensor, please collect the information described at http://linkwithlinux.blogspot.com/2013/01/crash-dump-analysis-installing-and.html. Note that you have to prepare the system for collecting crash dumps prior to the panic occurring.

6.3 Installation Verification

The following is a manifest of installed files:

Path	Additional Notes
/etc/init.d/cbdaemon	Sensor Daemon Script
/usr/sbin/cbdaemon	Sensor Daemon Executable
/lib/modules/$(uname -r)/kernel/lib/cbsensor.ko	Sensor Kernel Module
/etc/sysconfig/modules/cbsensor.modules	Kernel autostart file
/var/lib/cb/config	Binary store of settings file
/var/lib/cb/sensorsettings.ini	Settings file

To verify that the sensor daemon is running issue the following command:

```
pidof cbdaemon
```

There should be exactly one pid returned.

To verify that the sensor kernel module is running issue the following command:

```
lsmod | grep cbsensor
```

The output should show 1 item if the sensor kernel module is running.

6.4 Installation Failures

Installation are displayed to the screen. To check if the sensor is installed use the following command:

```
rpm -qa cbsensor
```

If the sensor is installed, then the a single line will be displayed showing version and build numbers:

```
cbsensor-v4.2.1.41002-1.x86_64
```

6.5 Sensor Communication History

Running inside terminal as root and sending the SIGUSR2 signal (via su):

```
      kill -n 12 $(pidof cbdaemon)
```

The log can be found at `/var/tmp/cb/sensor_comms.log`. Each transaction has a HRESULT (see description at closso@10.36.4.134:) which can be one of the following:

Facility number	**Description**	**Error code value**
203	OS level errors	Maps to errno
25	HTTP errors	HTTP error code
200	Curl errors	Curl error code (See `CURLcode` in `curl.h`)
201	Curl form errors	Curl form error code (See `CURLFormcode` in `curl.h`)

6.6 Manual Sensor Daemon Start & Stop

To restart the service, open a Terminal window and type:

```
sudo service cbdaemon restart
```

To start the service, open a Terminal window and type:

```
sudo service cbdaemon start
```

To stop the service, open a Terminal window and type:

```
sudo service cbdaemon stop
```

6.7 Clear catalog of observed binaries

In order to clear the existing catalog of observed binaries, open a Terminal window and type the following:

```
      sudo /etc/init.d/cbdaemon stop
      sudo rm -rf /var/lib/cb/store
      sudo /etc/init.d/cbdaemon start
```

6.8 Determine Server URL

To determine the server URL used by the sensor, follow the instructions in section 6.5 to create a communication log and dump the contents of the generated log file. The server URL appears at the top.

6.9 Trigger an immediate checkin to the server

Running inside terminal as root and sending the SIGUSR1 signal (via su):

```
kill -n 10 $(pidof cbdaemon)
```

6.10 Configuring core dumps

Next, add the following lines to /etc/sysctl.conf

```
      # Allow suid programs to dump core
      fs.suid_dumpable = 1

      # Dump core in /var/tmp
      kernel.core_pattern = /var/tmp/core
```

Finally, reboot.

Core dumps will now be automatically placed in /var/core/ suffixed with the pid of the crashing process.

Note: If a core file is collected for a bug report, please include the cbdaemon binary which crashed along w/ the core file.

6.11 Manual Core File Generation

Note: this section assumes the system preparation steps have been performed as described in the above Automatic core file generation section. To collect a core file for a live process (for example a process with high CPU utilization or appears to be hung) issue the following command:

```
	sudo gcore $(pidof cbdaemon)
```

A core file will be generated in /var/core/ and the process will continue as normal.

Note: If a core file is collected for a bug report, please include the cbdaemon binary which crashed along w/ the core file.

6.12 Driver Debug Parameters

Two arguments can be passed to the driver to control debug behavior:

g_traceLevel	Controls debug trace output flags See inc/dbg.h for specific flag values
g_eventFilter	Controls which event types are generated. See CB_EVENT_FILTER_* in inc/common.h for details on specific flag values.

These arguments can be passed either in the /etc/sysconfig/modules/cbsensor.modules file or by issuing the command ‘sudo insmod cbsensor.ko g_traceLevel=<value> g_eventFilter=<value>

insmod cbsensor.ko g_traceLevel=0x00200000

or

modprobe cbsensor g_traceLevel=0x00200000

0x00200000 - is hook tracing

- #define DL_INIT 0x00000001
- #define DL_SHUTDOWN 0x00000002
- #define DL_WARNING 0x00000004
- #define DL_ERROR 0x00000008
- #define DL_INFO 0x00000010
- #define DL_REQUEST 0x00000100
- #define DL_HOOK 0x00200000
- #define DL_VERBOSE 0x08000000
- #define DL_ENTRY 0x10000000

- #define DL_EXIT   0x20000000

^^^ are the available levels

Just OR them together to create the log level mask that you want

6.13 Daemon Debug option

While not ideal the daemon debug level and be raised by stopping the daemon and restarting in the following manner to get verbose logging:

```
/usr/sbin/cbdaemon info
```

6.14 Determine Sensor Version

To determine the version of cbdaemon running open a Terminal window and type:

```
cbdaemon -v
```

6.15 New Sensor INI settings

The Linux sensor has a new feature to allow the sensor to not record the process and file operations generated by a particular user. The new option is and must be placed in the sensorsettings.ini file and the data restarted. There is currently no way to remove the user settings from the kernel, other than deleting the INI setting and restarting the sensor kernel.

The UsersToIgnore setting takes a list of user names separated by colons, if multiple users are desired. The limit on users in the list is 5.

Single user:

```
UsersToIgnore=”username1”
```

Multiple users:

```
UsersToIgnore=”username1:username2”
```