Secure Endpoint Mac Connector Performance Tuning Guide

Contents

Introduction

Why do we need to tune?

Every time a file is created, moved, copied, or executed on a Mac endpoint an event for that file is sent from the operating system to the Secure Endpoint Mac connector. The event results in that file being analyzed by the connector. The analysis process generally involves hashing the file in question and running it through different analysis engines both on the computer and in the cloud. It is important to recognize that this act of hashing does consume CPU cycles.

The more file operations and executes that occur on a given endpoint, the more CPU cycles and I/O resources the connector will require for hashing. There are several features that have been added to the connector to reduce the overhead. For instance, if a file being created, moved, or copied has been previously analyzed, the connector will use a cached result. However, in the case of some events such as executes where security is paramount, all events are always fully analyzed by the connector. This means applications or processes that propagate multiple repetitive executions of child processes - especially over a short period of time - can cause performance issues. Finding and excluding applications that repetitively execute child processes at a rate greater that once per second can significantly reduce your CPU usage and increase battery life on laptops.

File operations such as creates and moves generally have less impact than executes, but excessive file writes and temporary file creation can result in similar problems. An application that writes to a log file frequently, or one that generates multiple temporary files can cause Secure Endpoint to consume a lot of CPU cycles with unnecessary analysis and can create a lot of noise for the Secure Endpoint backend. Distinguishing noisy parts of legitimate applications is a very important step in maintaining a productive and safe endpoint.

The purpose of this document is to help distinguish the file operations (create, move, and copy) and executes that will have a negative effect on the daemon's performance and waste CPU cycles. Identifying these file and directory paths will allow you to create and maintain the appropriate exclusion sets for your organization.

You can add pre-created exclusion lists to your policies that are are maintained by Cisco to provide better compatibility between the Secure Endpoint connector and antivirus, security, or other software. These lists are available on the Exclusions page in the console as Cisco-

Maintained Exclusions.

Types of Tuning

There are three kinds of exclusion tuning options available:

1. **Pre-Install Tuning – this can be done prior to installing the Secure Endpoint Mac connector. It will give you the cleanest look at which application and paths are the busiest on your machine. However, it's a very noisy process and requires the user to do a fair bit of analysis and aggregation on their own.**
2. S**upport Tool Tuning** – this can be done after the Mac connector is installed and can be performed on any endpoint without additional binaries. It performs a limited look back and is great for identifying troublesome applications.
3. **Procmon Tuning** – this process also requires the connector to be installed, but also requires the use of the Procmon binary, our custom tuning tool. It is essentially a more sophisticated version of the Support Tool Tuning feature. This method requires the greatest amount of configuration; however, it does provide the best results.

1. Pre-Install Tuning

Pre-Install Tuning is the most basic form of tuning and is done primarily through the command line in a Terminal session.

For newer mac from OS X El Capitan you will need to first boot to recover mode (command-r) while booting and disable the protection for dtrace:

```
csrutil enable --without dtrace
```

To inspect which file executions are most prevalent run the following:

```
$ sudo newproc.d | perl -pe 'use POSIX strftime; print strftime "[%Y-%m-%d %H:%M:%S] ", localtime'
```

This will generally show which applications are being run over and over again. Many provisioning applications will run scripts or execute binaries in short intervals to maintain company software policies. Any applications seen being executed at a rate greater than once a second, or executed multiple times in short bursts, should be considered a good candidate for exclusion.

To inspect which file operations are most prevalent, run the following command:

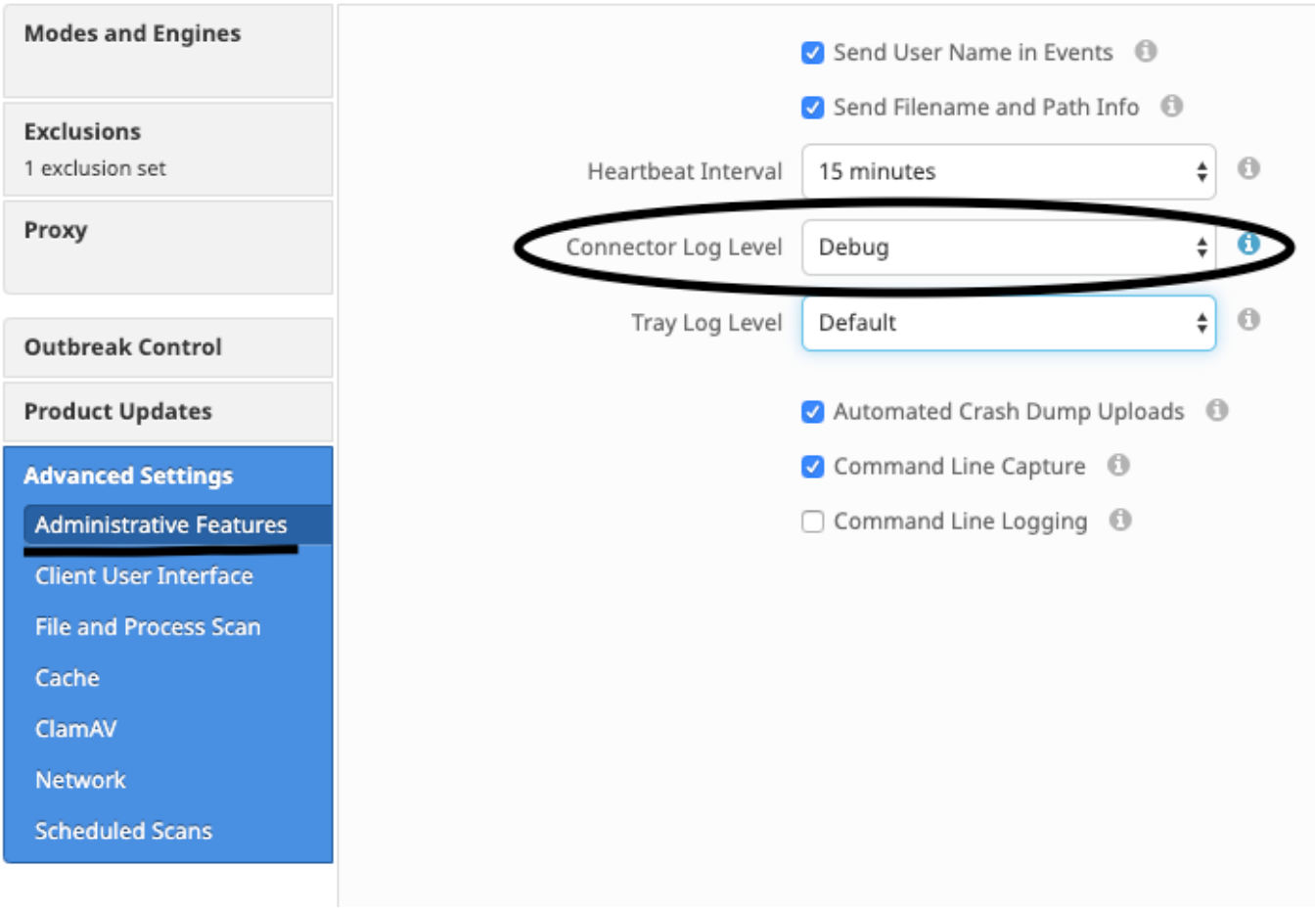
```
$ sudo iosnoop | perl -pe 'use POSIX strftime; print strftime "[%Y-%m-%d %H:%M:%S] ", localtime'
```

You will immediately see which files are being written to most. Often this will be log files being written to by running applications, backup software copying files, or email applications writing temporary files. In addition to this, a good rule of thumb is that anything with a log or journal file extension should be considered a suitable exclusion candidate.

2. Support Tool Tuning

Enabling Debug Logging

The connector's daemon needs to be put into Debug Logging mode before beginning support file tuning. This is done via the Secure Endpoint console, through the connector's policy settings at *Management -> Policies*. Select the policy, Edit the policy, and go to the *Administrative Features* section under the *Advanced Settings* sidebar. Change the *connector Log Level* setting to **Debug**.



Next, save your policy. Once your policy has been saved, ensure it has been synchronized to the connector. Run the connector in this mode for at least 15-20 minutes prior to continuing with the rest of the tuning.

NOTE: When your tuning is complete, don't forget to change the *connector Log Level* setting back to **Default** so that the connector runs in its most efficient and effective mode.

Running Support Tool

This method involves using the Support Tool, an application installed with the Secure Endpoint Mac connector. It can be accessed from the Applications folder by double-clicking on /Applications->Cisco Secure Endpoint->Support Tool.app. This will generate a full support package containing additional diagnostic files.

An alternative, and quicker, method is to run the following command line from a Terminal session:

```
sudo/Library/Application Support/Cisco/AMP for Endpoints/SupportTool-x
```

This will result in a much smaller support file containing only the relevant tuning files.

Either way you choose to run it, Support Tool will generate a zip file on your Desktop that contains

two tuning support files: fileops.txt and execs.txt. fileops.txt contains a list of the most frequently created and modified files on your machine. execs.txt will contain the list of the most frequently executed files. Both lists are sorted by scan count, meaning the most frequently scanned paths appear at the top of the list.

Leave the connector running in Debug mode for a 15-20 minute period, and then run the Support Tool. A good rule of thumb is that any files or paths that average 1000 hits or more during that time are good candidates to be excluded.

Creating Path, Wildcard, File Name, and File Extension Exclusions

One way to get started with Path Exclusion rules is finding the most frequently scanned file and folder paths from fileops.txt and then consider creating exclusion rules for those paths. Once the policy has been downloaded, monitor the new CPU usage. It might take 5 to 10 minutes after the policy is updated before you notice the CPU usage drop as it might take time for the daemon to catch up. If you are still seeing issues, run the tool again to see which new paths you observe.

- A good rule of thumb is that anything with a log or journal file extension should be considered a suitable exclusion candidate.

Creating Process Exclusions

NOTE: Process Exclusions on Mac can only be implemented for Mach-O files. Users cannot implement Process Exclusions for file formats such as .sh (Shell Scripts) or .app (Application Bundles).
For best practices regarding Process Exclusions see:Secure Endpoint: Process Exclusions in macOS and Linux

A good tuning pattern is first identifying the processes with a high volume of executes from execs.txt, find the path to the executable, and create an exclusion for this path. However, there are some processes that should not be included, this includes:

- General Utility Programs - It is not recommended to exclude general utility programs (ex: usr/bin/grep) without accounting for the following. The user can determine what application is calling the process, (ex: find the parent process that is executing grep) and exclude the parent process. This should be done if and only if, the parent process can be safely made into a process exclusion. If the parent exclusion applies to children, then the calls to any children from the parent process will also be excluded.The User that is executing the process can be determined. (ex: if a process is being called at a high volume by user "root", one can exclude the process, but only for specified user 'root", this will allow Secure Endpoint to monitor executes of a given process by any user that is not "root").**NOTE:Process Exclusions are new in connector versions 1.11.0 and newer. Because of this, general utility programs may be be used as a path exclusion in connector Versions 1.10.2 and older. However, this practice is only recommended when a performance trade-off is absolutely necessary.**

Finding the Parent Process is important for Process Exclusions. Once the Parent Process and/or User of the process are found, the user can create the exclusion for a specific user and apply the process exclusion to children processes, which in turn will exclude noisy processes that cannot themselves be made into process exclusions.

Identify Parent Process

1. From execs.txt, identify high volume process (ex: /bin/rm).
2. Open ampdaemon.log from support package, unzip syslog.tar, then follow path /Library/Logs/Cisco/ampdaemon.log (only available in a**full**support package, not from a support package generated with the default options).
3. Search ampdaemon.log for process to be excluded. Find the log line that shows the process execution (ex: Aug 19 09:47:29 devs-Mac.local [2537] [fileop]:[info]-[kext_processor.c@938]:[210962]: Daemon Rx: VNODE:EXECUTE X:6210 P:3296 PP:3200 U:502 [/bin/rm]).
4. Identify Parent Process using one of the following methods: Identify the Parent Process path which may follow the path of the process to be excluded (ex: [/bin/rm] [*Parent Process path*]).If the log does not include the Parent Process path, identify the Parent Process ID from PP: section of the log line (ex: PP:3200).
5. Using either the parent path or the Parent Process ID, repeat steps 3 & 4 to determine the parent of the current Parent Process. Continue this process until either no parent can be determined, or the Parent Process ID = 1 (ex: PP:1).
6. Once the process tree is known, look for the program path that covers most or all of the operations that should be excluded and uniquely identifies the application. This minimizes the chance of unintentionally excluding operations performed by another application.

Identify User of Process

1. Follow steps 1-3 of Identifying the Parent Process from above.
2. Identify User of a process using one the following method: `Find the User ID of the given process from U: in the log line (ex: U:502).From the Terminal window run the following command: dscl . list /Users UniqueID | grep #, where # is the User ID.You should see output similar to: Username 502, where Username is the User of the given process.`
3. This Username can be added to a Process Exclusion under the User category to reduce the scope of the exclusion, which for certain Process Exclusions, is important. **NOTE: if the User of a process is the local user of the machine, and this exclusion must apply to multiple machines with different local users, the User category must be left blank to allow the Process Exclusion to apply to all users.**