Web Gateway: Understanding Customized Logging and Log File Management

Log File Management

Introduction

This article explains the major aspects of how logging works inside the Web Gateway. After reading this article you will understand how the log handler and log management work on the Web Gateway, as well as how to create your own custom log. Modifying existing logs could also be extrapolated from this information.

Overview of Logging

A log entry is the result of the Web Gateway converting and formatting properties together while writing the result into a log file. At its very basic level, logging can broken into these two steps:

1. **Creating the User-defined log line**
2. **Writing the User-defined log line into a log file**

Below is a diagram that shows the flow of how properties are combined and then written into the log.

image-1615966580449.png

Here is what this process looks like in your Log Handler rules. (Policy > Rule Sets > Log Handler) The highlighted sections illustrate these two steps:

image-1615966589596.png

Creating the User-Defined log line

The first step in creating the log entry is to create an event to 'Set Property Value' and choose the User-Defined.logLine. From here, it's only a matter of choosing what properties to add, while converting, joining and formatting those properties to give your log file the look you desire.

Note: The access log must be formatted in a very specific way if it is being sent to Web Reporter or Content Security Reporter. Failure to adhere to that format will result in log parsing errors.
Other log files can be formatted as needed, for example if an alternate log parser requires a different log format.

Converting (casting)

Every property that you wish to write into a log file, needs to be of the data type 'String'. However, not all properties are of data type 'String' initially. If that's the case, it must be converted (cast) into a string. For example, you may wish to write the HTTP response code, which is a numerical value, to the user-defined log line. In order to log this property, it must be first converted into a string by the 'Number.ToString' property.

Note: what proceeds the 'ToString' is the data type that is being converted to a string. This is how it looks:

Number.ToString (Response.StatusCode)

Another common property that you'll also need to convert (cast) is the 'Client.IP' property. This property uses the data type IP and also needs to be converted to a string. This is how it looks:

IP.ToString (Client.IP)

Joining/Formatting properties

Since multiple properties will usually be used to create each log entry, the joining all of these properties is the next important item to understand. While it's actually handled by a process called concatenation, for the remainder of this document we will refer to it as 'joining' for simplicity.

The joining process begins at the start of the User-Defined log line you are creating. To begin, you'll just need to start adding the properties you wish to log using the Converting (casting) method we discussed above. Then, you'll want to add a Fixed String to separate or format the Properties so that the logline gives you the look and feel you desire.

The flow of this would look like:

User-Defined Logline = **<Property>** + **<Fixed String>** + **<Property>** + **<Fixed String>** + **<Property>** + **<Fixed String>**

The <Fixed String> can be any form of free text and is often used as a delimiter between the Properties you want to log. Examples could include quotations, spaces, other delimiters, or just descriptive text.

For example, below is a sample rule to show what this looks like in your Log Handler rules:

image-1615966615814.png

In the example above you can see that the highlighted areas are the 'Fixed Strings' that are used. In this example we used spaces and quotations. The properties and Fixed Strings added together would look like:

Set User-Defined.logLine = DateTime.ToWebReporterString "Authentication.Username" "String.ReplaceIfEquals(IP.ToString(Client.IP),"","-")" "List.OfString.ToString(Antimalware.VirusNames)" "URL"

This line when written to a log file would output like:

[27/Mar/2013:00:21:03 +0000] "mjordan" 10.10.1.1 "McafeeGW: EICAR test file" "www.eicar.org/download/eicar.com.txt"

If the line doesn't output like you expect, you know you need to modify the properties and/or Fixed Strings to get the look you desire.

Dealing with potentially empty values

Since both log parsers and people reviewing the log files could get confused by an empty value, it can sometimes be important to replace an empty field with a pre-selected value. When needed, the default logs will replace empty values with a dash character, but it could be virtually anything. One example for a value that could potentially be empty is the username field. Not every request is authenticated (due to whitelisting or simply because it has not been authenticated yet). To not leave a "gap" in your log file line, you might want to replace this non existing username with a dash or even another string like "no-user".

The property that calls this is String.ReplaceIfEquals, and it requires three additional strings to be specified to work. The first is what string is being checked, the second is what value is being looked for, and the last is what value should be used for replacement.

Here is an example that replaces and empty username with a dash:

String.ReplaceIfEquals(Authentication.Username,"","-")

Writing the User-Defined log line

The last step is to send the log string to the File System Logging Engine that manages the logs, which is accomplished by the event FileSystemLogging.WriteLogEntry. This event uses the user-defined property that stored the string as a parameter, and sends it to a specific file system

logging engine.

There is a more detailed example of creating your own log later on that will expand on everything covered in this section.

Overview of the File System Logging Engine

The duty of the File System Logging engine is to not only name log files and write lines passed to it from the log handler, but also to manage them, for example in regards to rotation and deletion. A file System Logging Engine can be configured under 'Policy > Settings > Engines > File System Logging'. Below, we'll discuss some of these management settings.

General Engine Settings

Name of the log

In the space provided, specify a name for your logfile:

Example: troubleshooting.log

Note: Once you've successfully created a log file, the file itself can be found under 'Troubleshooting > Log Files > User-defined-logs'

Enable Log buffering

When selected, the log is buffered before being written with its default interval of 30 seconds. **It's recommended to have this enabled as it reduces disk I/O and improves performance.**

Enable header writing

This creates a header line that appears at the top of every log file to denote what properties are contained within the log. When configuring or modifying a log, it is important to match these header values with the properties that are written into the logs by the log handler.

Note: This is very important for log files that are being used by Web Reporter. Failure to match the header and formatting to the properties being logged will result log parsing errors.

Log Encryption

Allows log files to be encrypted requiring one or two passwords to decrypt it. Use if needed.

Rotation, pushing, and deletion

Any log created by the Web Gateway should have appropriate rotation, deletion, and push schedules set. Without proper log management, logs will continue to use up more and more space and eventually cause disk space issues on the /opt partition. For more information on monitoring this partition, please see this community article: https://community.mcafee.com/docs/DOC-4824

There are global log management settings found under 'Configuration > Log File Manager', however for user-defined logs such as the access.log, access_denied.log, etc we recommend that you configure these settings per log engine.

Note: When a File System Logging engine is configured, its settings take precedence over those configured in the Global settings.

Rotation

When a file reaches the specified limit (...if file size exceeds), a new log file will be created, and the old file will now reference a file name that includes the date and time of when it was rotated. For example, if you rotated your access.log at 10:30am on February 1, 2013, the additional information is broken down into yymmddhhmm so the rotated log will be access1302011030.log. By default, logs are rotated daily at midnight, but an interval can be implemented as needed based on hours and minutes from the last rotation time.

Note: It is recommended to enable the 'GZIP files after rotation' setting as it will help save disk space on the /opt partition.

Deletion

This can be set based on either the total number of the specified log, or how old it is, and triggered based on whichever comes first.

Pushing

Files can be pushed using FTP, HTTP, HTTPS. The user name field can also contain %h which is replaced by the host name of the proxy, which can be useful to differentiate the source of the given log. The pushing interval can either be done directly after rotation, or based on hours and minutes from the last push. Additionally a next hop proxy can be configured here if needed.

Creating a Customized Log

The ability to create your own custom logs can not only assist in overall data collection, but can help administrators troubleshoot an issue. This example will cover the second, and assist in

tracking down requests from a specific user that has been shown to make a large number of either bad requests or a lot of generally blocked traffic. Keep in mind that this criterion can easily be modified once the log is in place, and simply enable/disable the log when data is needed.

Custom Log - Making the rule

To begin the process, select the Default log handler and select Add > Rule Set, which brings you to the following:

image-1615966646164.png

Name it what you wish but try to make it easy to understand for anyone else that might also need to administrate the Gateway. This example will be called Troubleshooting Log. As suggested previously, criteria can be added in the rule set or within the rule itself, this one will be done within the rule. The next step is to add a rule to this new log, so select Add Rule which brings you to the following:

image-1615966655234.png

After naming, select next. Then select Add > Advanced Criteria. Since the concern is a specific user and their IP is known, the first criteria is going to be Client.IP, which will be shown in the next screen shot, but additionally the same criteria that’s used for the access_denied.log is going to be added as well. Note that using small portions of the properties names can be used as a filter to speed up finding the property you want.

image-1615966666310.png

With the additional criteria, it will end up looking like the following:

image-1615966678329.png

Note that the Criteria Combination was modified so the logic will always expect the Client.IP or what would normally be the criteria for writing the access_denied.log. The next step is to set an action, but generally all logging rules will have a continue action, which is also the case for this one. The next step is to setup the Events.

Custom Log - Creating the User-Defined log line

The layout will look just like the default logs, where a user-defined property is used to store the log line, properties will be used to fill it, and the FileSystemLogging.WriteLogEntry will call the settings engine to handle the rest.

Select Add > Set Property Value > and in the dropdown, select User-Defined.logLine as seen below:

image-1615966687387.png

Here is where the remainder of the layout is up to you. This example will have the following order, using the actual property values listed:

- **DateTime.ToWebReporterString**
- **Authentication.UserName**
- **Client.IP**
- **Request.Header.FirstLine**
- **Header.Request.Get with a parameter of user-agent**
- **Rules.CurrentRuleSet.Name**
- **Rules.CurrentRule.Name**
- **Block.ID**
- **Block.Reason**

What will be covered next is how to add a property, then how to add a delimiter, followed by using String.ReplaceIfEquals on the property Authentication.UserName, then another delimiter before covering converting using the Client.IP property

Continuing from the previous screen shot, select Add below ‘To concatenation of the strings:’ and then select the Use Property box, which then allows a filter to be used to assist in finding the property you wish to add, as seen below:

image-1615966709493.png

I typed web to find the first property desired for this log, which is DateTime.ToWebReporterString. Once selected, choose OK to return to the Edit Set Property window so the next piece of the log entry can be added. Since a delimiter is needed to avoid having these values appear all mashed

together, that will be next. Just as before, select Add, but instead of choosing Use Property, a string can be entered by typing it into the Parameter Value window. A delimiter should be picked based on what makes the logs easier to review, so the same method will be chosen that is already in use on the Gateway, and will also assist in explaining what was converted in the Formatting section of this article. Since the previous value does not make use of the quotes, what should be put into this field is a single space, and then a double quote to start surrounding the next property in double quotes. What this accomplishes is to have a single space after the previous property, and begins the following property with a double quote, and since the next property will now begin with a double quote, the delimiter following that property will start with a double quote to properly surround it. Here is how the first delimiter value will look:

image-1615966719111.png

The next property being added is Authentication.UserName. Since there are times that this can be blank, using the property String.ReplaceIfEquals will be shown here. Unlike the first property that's already a string and is selected first, Authentication.UserName is not the first property selected and this time it will be String.ReplaceIfEquals. Here is how this is accomplished:

image-1615966728963.png

The parameters for String.ReplaceIfEquals must be modified to what is seen above. As mentioned before, the first parameter is the string that's being checked, the second is the string that is being search for, and the last is what the second string will be replaced with. So, based on what was added, Authentication.UserName was the string (property) searched, and if it's blank, it will be replaced with a dash.

Now, to finish surrounding this property in double quotes, here is how the next delimiter is setup. Since the next property is going to be Client.IP and the rule is not accounting for empty values, it's a good idea to surround it in double quotes as well, so this will be a double quote, followed by a space, and then a double quote. Here's how that's setup:

image-1615966738703.png

The next property that will be added is Client.IP and since this is a data type of IP on the Gateway, it must be converted into a string before it can be written into the log line. Here is how that's accomplished:

image-1615966747323.png

The only other property with any special handling not already covered Header.Request.Get. This also makes use of parameters, but this time instead of using a property, the value that's being retrieved is typed in instead, somewhat like a delimiter. In this example, the user-agent is being gathered from the header information, and also keep in mind that this is not case-sensitive. Here's how to get the user-agent:

Note: Since this is added to the log after Request.Header.FirstLine, that property will be seen in the screen shot as already being added even though the steps to add it were not covered.

image-1615966757042.png

Custom Log - Writing the User-Defined log line

The remainder of the properties being added into the log line will not be covered by screen shots since they do not require any different, or additional instruction, that have not already been covered. The next step is to send this log line to a file system logging engine that handles writing the new entry to the log it manages. This is accomplished by adding a new event called FileSystemLogging.WriteLogEntry. Here's how to add that event, as well as define that it uses the user-defined.logLine that was being created up to this point.

image-1615966777419.png

Now, after the logLine has been chosen, a new file system logging engine must be created to manage this, so after setting the parameter as seen above, here's what's done next:

image-1615966786736.png

The entire log header cannot be seen in the screen shot, so here is the full line:

time_stamp "auth_user" "src_ip" "req_line" "user_agent" "rule_set" "rule" "block_res" "block_page_res"

The next step is to enable and configure your rotation, deletion and pushing settings. The screen shot will explain what was changed from the initial settings, and the best practices already covered will be configured.

image-1615966797082.png

This is how the completed logging rule looks like in the log handler.

image-1615966808940.png

You've now completed creating your custom logging rule. Any custom log can also be referred to as a 'User-defined-log. These logs can be found under "Troubleshooting > Log files > User-defined-logs ".

Link

https://community.mcafee.com/t5/Documents/Web-Gateway-Understanding-Customized-Logging-and-Log-File/ta-p/554106

Revision #1
Created 17 March 2021 07:34:23
Updated 17 March 2021 07:40:41