Log Normalization Systems and CEE Profiles

Rainer Gerhards, Adiscon GmbH

2011-10-24

Draft 1.0.1

Overview

This document describes the overall approach of log normalization and its relationship to CEE and CEE profiles.

Definitions

Log Normalization

For purposes of this document, "**log normalization**" is the process of transforming original log messages from different formats into a desired target format, for example a CEE (see http://cee.mitre.org) event record. Different types of normalization are discussed below.

The common task is that the input formats need to be transformed into a single target format. It does not so much matter which formats are source and target.

Format Structuredness

- Semi-structured (like in Linux log files, no clear distinction between field values, field delimiters and noise data)
- Weakly structured (like CSV-based formats, needs external information to understand fields, field values are provided)
- Strongly Structured (like RFC5424 structured data, field names and values are provided)
- Strongly structured based on a validating profile (like a CIM event, field names, values and semantics are provided)

Rule Base

The term "**rule base**" means any set of information that is designed to guide the transformation process. A rule base consists of multiple rules. Practical implementation of rule bases are considerable different. Among others, they may be a simple field list for CSV-based messages or a set of regular expressions describing semi structured messages.

Transformation Layers

Transformation may be done on a purely syntactical layer or on a semantic layer.

A syntactic translator

a) Reads the input message

b) Extracts fields and field names from message using rule base

c) Transforms the message in the desired target format. This includes two functions:

    a. At a minimum, the target format has proper field name and field value syntax, including required character set (EBCDIC-to-ASCII and big/little endian conversions fall into this category as well as reformatting the way name-value pairs are expressed).

    b. Normalization of field values themselves may be required. Often, different encodings use different values for otherwise-identical objects. A standards-based example for such field value translation is RFC5674, sect. 2, which specifies how to translate syslog severity into an ITU perceived severity. The key point to differentiate syntactic vs. semantic translation is that field value translation as specified here is based on mapping tables, without the need to fully understand the event or the property (but obviously this is a border case that one could argue requires some very limited understanding of semantic).

A semantic translator seems to add “just” an additional layer of transformation. However, its logical processing is actually somewhat different:

1. Read input message

2. Extract fields and field names from message using rule base

3. Identify the exact meaning of the event (e.g.is it a logon or logoff event)

4. Transform the message to the desired target format. This includes:

    a. Use the target format’s proper record format and character encoding

    b. Normalize field values

    c. Ensure that all fields required by the target format are present and drop fields that are present in the source but cannot be represented in the target. Note that this processing step may require massive modification to the message itself. The source message may explicitly contain only a subset of the required target information. If so, the translator must obtain the missing information either via the rule set or implicitly from the other source message properties. It may even be impossible to do a proper transformation due to missing source message content or other source and target profile incompatibility (e.g. flat vs. structured message content!).

In summary, a syntactical message translator actually works on strings of characters, which are transformed using (relatively) simple rules. A true semantic translator instead works on what the event

actually represents. It needs to “understand” the source event so that it can convey the right information, potentially in very different terms, to the target format. The latter is obviously much more complex.

However, there is a hybrid form, a kind of relaxed semantic translator. It operates mostly like a pure-syntactical translator, but is able to augment some information with the end result that a proper semantic conversion is done without deep understanding of the event itself. My assumption is that most of current translators can be converted in such a relaxed semantic translator.

In the following, I will describe in details how a syntactic translator processes messages and how it can be converted into a relaxed semantic translator.

Translating messages

In order to transform messages correctly, a translator needs to parse the input message and extract field names and field values. For this, it needs to know

1. Field names – for strongly structured data usually contained in the message or accompanying profile or obtained via the rule base. For simplicity, we assume structured field names in case of structured events, that is the structure is conveyed via the field name.

2. Field syntaxes – This may include conversion rules for field value conversion. Usually obtained either from profile information or the rule base.

3. Parsing rules for obtaining fields from the message. These may be format-induced or provided by the rule base. For example, for CSV this may be a list of field names and field order. For semi-structured formats this may be regular expressions describing the format. For strongly structured and standardized formats, everything may be provided by the standards documents.

4. If field value conversion is required, mapping rules for values, on a per-field-basis, must be provided. This can typically be implemented as side-tables inside the rule base.

The translation process also tends to require a sense of the message’s semantic. This information may not explicitly be recorded, but is implicitly present. We describe this by format type:

- Semi-structured formats: the rule base must contain rules to uniquely identify messages. Exact matches are required, otherwise messages may be misclassified. In order to obtain exact matches, the format of a message must be known very well. This carries the co-notation that the rule base creator also knows the meaning of the message. This knowledge may not be recorded in the rule base, but is nevertheless available at rule base creation. So it could be augmented to the rule base. Note that some messages may actually have the same format to differentiate between two semantic events. For example, a logon and logoff event may come with mostly the same semi-structured message containing mostly the same fields. It may be discernible only by a specific field value that indicates either a logon or a logoff. Now for the contrary assume that no such discerning property is available. In that case, the message cannot be classified in terms of

logon/logoff at all. As such, it is a different semantic class of message, what then can be recorded[1].

- Weakly structured formats: here the sense of semantic is on a per-logfile basis. Usually weakly structured formats like CSV contain records of a single event type within a log file. This correlates to the fact that number and position of fields is usually fixed. As such, the event type designator can be applied on a global basis to the rule base as whole, again assuming that the rule base creator knows what he deals with (a hopefully valid assumption). In cases where weakly structured formats actually represent different event types, the actual type could be identified on some field value. However, it is definitely possible that correct classification is not known and so event type assignments cannot be made. We consider this to be a rarely occurring case.

- Strongly structured formats: here, it is actually less probable that event type information exists. If there exists a standard document for the format (e.g. like with parts of syslog's structured data fields), event type information can be obtained from there. On the other hand, it is very likely that no or a very limited rule base needs to be created for conversion, because the syntax of strongly structured formats is well known. As such, there is a slighter chance of a human looking at the actual messages. As a result, correct event type classification may not be known.

- Strongly structured formats with profile information: While it is unlikely that a classifying rulebase exists, the profile documents should usually provide sufficient information to classify messages by event type.

As can be seen, event type information is expected to be present in most cases, except for the important case of strongly structured formats (without profile information). Excluding that case and assuming that implicitly-known event type information is added to the rule bases, a simple syntactic translator can be relatively easy extended to a relaxed semantic translator.

For this, steps one to four above of the purely syntactical translation are carried out. Then, additional steps are added before writing the target format:

5. Based on event type and target profile, verify that the all information is present to create a well-formed target event record. Note that in this step insufficient information may be detected. The translator needs to decide how to handle that. The obvious options are to either ignore (drop) the event or create an error-type of event.

6. Augment the event information with the event type, completing the target data set (for CEE this means adding the proper profile and event id).

After this has been done, the target event can be written. Note that in this process, it is assumed that most input fields can be converted with very simply translations (like lookup-tables and simple copy/re-encoding) to the output format. In contrast to a full blown semantic translator, no complex translation logic on multi-field level (or even multi-record level) is done.

[1] This issue touches on the topic of „flat-vs.-thin" logs and consolidating event log records over multiple input log records. A topic currently not to be addressed in this paper.

Given the current state of the art, including existing implementations, it is assumed that this relaxed semantic translation can be done in high speed and without prohibitive implementation cost. As such, it is an attractive solution to make existing legacy log data compliant to the target format.

Impact to CEE

As has been described, translators usually require both source messages as well as translation rules, usually given in form of rule bases or similar concepts. The translator usually tries to uniquely identify the event. The information used for this can easily be augmented by matching the CEE profile ID. With that additional information, translators can create syntactically and semantically correct CEE log records.

Some source messages do not contain all information required by a CEE profile. If that information cannot reliably deduced, the message is simply non-conformant and must be treated as such.

All in all, the CEE profile requirement does in my opinion, in most cases, not create considerable extra complexity for translators.

Credits

Thanks to Raffael Marty for reviewing versions of this document and providing valuable feedback.