Simplifying Correlation Rule Creation for Effective Systems Monitoring

C. Araujo[1], A. Biazetti[1], A. Bussani[2], J. Dinger[1], M. Feridun[2], and A. Tanner[2]

[1] IBM Software Group, Tivoli Raleigh Development Lab, Raleigh, NC 12345, USA
{caraujo, abiazett, jd} @us.ibm.com
[2] IBM Research, Zurich Research Laboratory, 8803 Rueschlikon, Switzerland
{bus, fer, axs} @zurich.ibm.com

Abstract. Event correlation is a necessary component of systems management but is perceived as a difficult function to set up and maintain. We report on our work to develop a set of tools and techniques to simplify event correlation and thereby reduce overall operating costs. The tools prototyped are described and our current plans for future tool development outlined.

Event correlation is a key component of systems management. Events from multiple resources, e.g., network elements, servers, applications, are collected and analyzed to detect problems such as component failures, security breeches and failed business processes. Management solutions require correlation for filtering and analyzing massive numbers of events, for example by removing duplicate events, or for detecting event sequences that signal a significant occurrence in the managed systems. Relevant event patterns need to be identified and formulated as rules, and mechanisms provided to map observed events into the defined patterns. Many systems allow correlation of events where the patterns are expressed as rules [1]. The difficulty lies in identifying the different and relevant patterns of events, as patterns change and new ones are introduced. Our goal is to develop tools to help operators and systems management architects to identify event patterns, to create rules to implement the patterns, test their validity, and to monitor and manage the rules during their lifecycle.

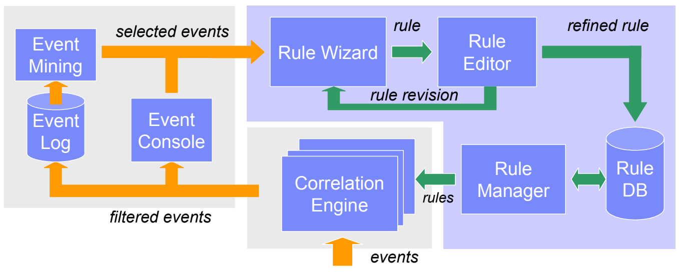

Fig. 1. Tools for automated correlation rule generation

The tool collection is shown in Figure 1. The *correlation engines* use installed rules to filter incoming events, which are logged (*Event Log*) and displayed on the *event console*. By *event mining* [2, 3] or operator intervention, patterns of events that need to be filtered or in sum indicate a situation are selected. In the *rule wizard* and the *rule editor*, the patterns are used as input to create rules, possibly through several refinements, stored in the *rule database*, and distributed for deployment to the relevant correlation engines.
Here, we describe the first stage of our work focusing on the prototype developed to create rules based on events selected by an operator from an event console. The

automation of the rule-generation process begins with the operator selecting a number of related events from the event console, for example false positives that should be filtered out. First, the operator invokes the rule wizard which allows him to select a pre-defined rule pattern to apply to the selected events. The prototype offers six patterns: *filter* to match an event against a predicate; *collection* to gather matching events within a time interval; *duplicates* to suppress duplicate events by forwarding the first matching event and blocking similar events until the end of the time interval; *threshold* to look for a sequence of similar events crossing a threshold value within a time interval; and *sequence* to detect the presence/absence of a sequence of events (in order or random) within a time interval. Second, the operator can select the parameters relevant to the selected pattern, e.g. the time interval during which the pattern should occur. The third step generates the predicates used in selecting pattern-relevant events. The operator is presented with the attributes available in each event and selects the one to be used in the predicate to filter the incoming events. With this information, the wizard automatically generates a predicate expression for the rule. Finally, the operator specifies actions to be executed when the rule triggers, i.e., detects the defined pattern. Actions can include updates to the events (e.g. relating to another event, changing attributes of an event, closing/acknowledging an event) or sending notifications (paging, email) to interested parties among others.

An example application of the rule wizard for a fax server demonstrates the usefulness of the approach. In the case of a failure, a number of related events, such as *rfboard_down* and *rfqueue_down* are observed by the operator at the console. From experience the operator knows that these are related to a defective fax server and decides to create a rule to collect and summarize them into one event. He selects the related events at the console and invokes the rule wizard, which shows the selected events and the rule pattern options. The operator chooses the sequence pattern, configures rule parameters, e.g., the time window, selection attributes, e.g., event type, and selects an action to summarize all selected events into a single *nt_rfserver_down* event. The rule is automatically created, and once deployed will result in just one summary event displayed on the event console, instead of the multiple original ones.

We have developed and demonstrated a prototype rule wizard and rule editor, and integrated it with the IBM Tivoli Event Console (TEC). The prototype enables automatic creation of rules, which greatly helps operators by providing a simple way to create rules based on observed events. Our follow-on work and areas of investigation focus on the development and integration of tools for testing and debugging newly created rules–for example checking how new rules interact with the existing deployed rule sets—using the current event stream, historical event logs or simulated event flows as an additional refinement step prior to rule deployment in the real environment. The rule wizard can be extended to define frequently occurring and typical high-level tasks of an operator, such as 'filter these events out' or 'page administrator when these events occur together' as templates for use by less skilled operators. We also address instrumentation of the correlation engine to collect real-time data on rule behavior and performance as feedback into rule design and improvement, and for the management of the lifecycle of the rules.

References

[1] IBM Tivoli Event Console, http://www.ibm.com/software/tivoli/products/enterprise-console/.

[2] J. L. Hellerstein, S. Ma and C. Perng, "Discovering actionable patterns from event data", IBM Systems Journal, vol. 41, issue 3, pp. 475-493, 2002.

[3] K. Julisch, "Clustering Intrusion Detection Alarms to Support Root Cause Analysis", ACM Transactions on Information and System Security, 6(4):1-29, 2003