SEC – a Lightweight Event Correlation Tool

Risto Vaarandi

SEC – a Lightweight Event Correlation Tool

Risto Vaarandi
Department of Computer Engineering
Tallinn Technical University
Tallinn, Estonia
risto.vaarandi@eyp.ee

***Abstract*—Event correlation has become one of the most important techniques in today's network management, and there is a clear trend to extend its use to other application domains as well. Unfortunately, existing event correlation systems are often platform-dependent and heavyweight solutions that have complicated design, being therefore difficult to deploy and maintain, and requiring extensive user training. Their complexity and size makes them often unfeasible to apply for smaller networks and for smaller event correlation tasks. Also, some systems are cumbersome to use outside the domain of network fault management. In addition, commercial event correlation products tend to be quite expensive. In this paper the author presents a lightweight, open-source, and platform independent tool for rule-based event correlation called SEC (Simple Event Correlator), and describes its application experience.**

Keywords—event correlation

I. INTRODUCTION

Event correlation has become one of the most important techniques in today's network management. Without applying this technique, the arrival rate of network events often becomes too high for the human operator to follow the changes in the network state. The employment of event correlation allows one to reduce large amounts of network events to smaller and more meaningful sets of alarm messages that can be handled by the human operator in a timely manner.

Although currently event correlation has been mainly used for network fault management, there is a clear trend to extend its use to other application domains as well [1], most notably to security management and intrusion detection. Staniford et al. have implemented Spice event correlation engine for detecting stealthy portscans [2], GrIDS [3] uses graph-based event correlation for detecting intrusions, Snort [4] applies event counting for detecting portscans, etc. Event correlation could not only be used in the intrusion detection process itself, but also as a postprocessing technique for IDS output, in order to provide the human operator with a concise view of security alarms.

Event correlation techniques have also been proposed for system administration and logfile analysis. Some logfile monitoring tools like Swatch [5] and Logsurfer [6] support a few event correlation operations, but their event correlation capabilities are nevertheless limited. A recent paper by Bing and Erickson [7] points out the weaknesses of the current logfile monitoring tools, and discusses the importance of extending the logfile monitoring techniques with more complex heuristic approaches, including event correlation.

Although event correlation systems that are currently available on the market (like HP ECS [8], SMARTS [9], and NerveCenter [10]) have been highly successful and are used worldwide by many larger companies, they suffer from a number of drawbacks.

Firstly, existing systems are often heavyweight solutions that have complicated design and user interface. This means that their deployment and maintenance is time-consuming, and they require extensive user training. Also, their complexity and size makes them often unfeasible to apply for smaller networks and for smaller event correlation tasks, especially on nodes with limited hardware capabilities (e.g., a logfile monitoring and correlation on a workstation with small disk, little memory, and weak CPU).

Secondly, since existing systems are mostly commercial, they are platform-dependent - customers are supplied with program binaries that run on a limited number of operating systems. Furthermore, several commercial systems have been designed for one particular network management platform only. Some systems also suffer from the fact that they have been designed specifically for network fault management, and their application in other domains (e.g., logfile analysis) is cumbersome.

Thirdly, existing systems tend to be quite expensive. Therefore, many academic institutions and smaller companies with more limited budget are unable to use them for daily network management tasks or for research experiments. Since a lot of research has been done in the field of event correlation recently, some experimental correlation engine prototypes have been created, but most such prototypes are not publicly available on the Internet. Although many excellent open-source network management solutions exist [11], there is no freeware correlation engine available yet which would be mature enough for use in a production environment. The OpenNMS [12] team plans to implement MAJI correlation engine after first versions of OpenNMS have been released, but currently only the code specification of MAJI is available. Another event correlation related open-source project is CLIPS [13], which is an environment for creation of rule-based expert systems. Although CLIPS itself is not an event correlation tool, it has been successfully used for constructing event correlation systems [1, 14].

For the reasons above, quite many sites are using homegrown event correlation solutions, which often comprise

This work was supported by the Union Bank of Estonia and by Estonian Science Foundation (Grant No. 4067).

of a few application-specific shell scripts. Each time a new application is set up, a new solution has to be developed, which is rather impractical and time-consuming.

In this paper the author presents an open-source platform independent tool for rule-based event correlation called SEC (Simple Event Correlator), and describes its application experience. The first versions of SEC were applied for network fault management [15], but by now SEC has evolved into an event correlation tool that is used in other application domains as well, such as intrusion detection, logfile monitoring, fraud detection, etc. The primary design goal of SEC was to fill the gap between homegrown and commercial solutions, and to create a lightweight and easily customizable tool that could be used for a wide variety of event correlation tasks, either standalone or integrated with other applications.

II. SEC DESIGN

SEC is an open-source event correlation tool that uses rule-based approach for processing events. Its main design objectives were platform independence, lightweight build and simple configuration, applicability for a wide variety of event correlation tasks, and low consumption of system resources.

To achieve independence from operating system platforms, the author decided to write SEC in Perl. Since Perl runs on almost every operating system flavour and has become a standard part of many OS distributons [16], applications written in Perl are able to run on a wide range of operating systems. In addition, Perl programs are almost as fast as programs written in C.

SEC does not need much disk space and is very easy to install, since its current size is only about 160KB, and its configuration is stored in regular text files that require only few kilobytes more. Also, since SEC is written entirely in Perl, it can be used instantly after its source distribution has been unpacked, without any additional preparations (such as compiling and linking the source).

SEC receives its input events from a file stream, and produces output events by executing user-specified shell commands. Regular files, named pipes, and standard input are currently supported as input, allowing one to use SEC with any application that is able to write its output to a file stream. Applications that have an event management API can also be integrated through simple plugins that employ API calls to read the application's event stream, and copy it to the standard output or file (a sample plugin for HP OpenView ITO is part of the SEC package).

To be able to handle input events regardless of their format, SEC uses regular expression language for recognizing them. That eases the configuration of SEC, since many UNIX tools (like *grep*, *sed*, *find*, etc.) are relying on regular expressions, and therefore most system and network administrators are already familiar with the regular expression language.

SEC configuration is stored in text files which can be created and modified with any text editor. Each configuration file contains one or more rules, and rulesets from different files are applied logically in parallel. In addition to event matching condition, most rule definitions specify a list of *actions*, and optionally a Boolean expression of *contexts*. The SEC contexts represent the knowledge that SEC has learned during the event correlation process, with each context having a certain lifetime (either finite or infinite). Contexts can be used for activating and deactivating rules dynamically at runtime, e.g., if a rule definition has (X OR Y) specified for its context expression and both context X and context Y don't exist at a given moment, the rule will not be applied. Another important function of the SEC contexts is to act as event stores – events of interest can be associated with a context, and all the collected events supplied for an external processing at a later time.

Currently, SEC supports the following rules types:

- *Single* – match input event and execute an action list.
- *SingleWithScript* - match input event and execute an action list, if an external script or program (e.g., query to a network topology database) returns certain exit value. The external script or program will be supplied with the names of existing contexts through its standard input.
- *SingleWithSuppress* - match input event and execute an action list, but ignore following matching events for the next t seconds.
- *Pair* - match input event, execute an action list immediately, and during the next t seconds ignore following matching events until some other input event arrives. On the arrival of the second event execute another action list.
- *PairWithWindow* - match input event and wait for t seconds for other input event to arrive. If that event is not observed within the given time window, execute an action list. If the event arrives on time, execute another action list.
- *SingleWithThreshold* - count matching input events in the window of t seconds and if a given threshold n is exceeded, execute an action list. The window is sliding.
- *SingleWith2Thresholds* - count matching input events during t seconds and if a given threshold n is exceeded, execute an action list. The counting continues after the execution - when no more than n' events have been observed during the last t' seconds, another action list will be executed. Both event correlation windows are sliding.
- *Suppress* - suppress matching input event.
- *Calendar* - execute an action list at specific times.

SEC actions were not only designed for generating output events, but also for making rules to interact, for storing and managing knowledge, and for connecting external fault or knowledge analysis modules to SEC. The following actions are currently supported:

- *none* – no action
- *logonly* – log a message

- *write* – write a line to a file or a named pipe
- *shellcmd* – execute an external shell script or program
- *pipe* – execute an external shell script or program, and feed data to its standard input
- *spawn* – execute an external shell script or program that will provide SEC with additional input events (e.g., run *tail –f* on a file, start a SEC subdaemon for topology-based event correlation, etc.)
- *create* – create a context, and optionally set some of its parameters (e.g., lifetime) to non-default values
- *set* – set the parameters of a context to new values
- *delete* – delete a context
- *add* – associate an event with a context
- *report* – supply all events of a given context for an external processing
- *event* – generate a new input event that can be matched by other rules
- *reset* – cancel an event correlation operation (e.g., reset ongoing event counting)

By combining several rules with appropriate action lists and context expressions, more complex event correlation schemes can be defined. Fig. 1 presents a SEC ruleset example.

When a SEC rule matches an input event, SEC either starts a new event correlation operation for the recognized event, or lets the event be correlated by some event correlation operation that is already running. When SEC starts an event correlation operation that can't be completed immediately (e.g., the operation involves correlation over a time window), it stores the operation in its working memory. The other entities kept in the working memory include rules, contexts, data about running child processes, etc. Most SEC internal data structures are implemented as Perl hashes, which can be searched rapidly, even when the hash is very large.

Fig. 2 depicts the work of a sample SEC ruleset. If input event *"Interface X at node Y down"* has been observed and no *interface up* input event will be received for the same interface within 15 seconds, event *"DOWN_X@Y"* will be generated; otherwise event *"BOUNCE_X@Y"* will be generated. When event *"DOWN_X@Y"* has been observed, output event *"X@Y down"* is produced; when input event *"Interface X at node Y up"* arrives subsequently, output event *"X@Y up"* will be produced. When event *"BOUNCE_X@Y"* is received, a counting operation is started for the event; also, an external fault analysis module is spawned that monitors the interface X for line-level faults during 1 hour. If the analysis module detects a line-level error, the counting operation is cancelled and output event *"X@Y line fault"* is produced. Also, the context *LINE_ERR_X@Y* will be set up for the next 5 hours, in order to avoid possible repeated *"X@Y line fault"* output events. If the analysis module does not detect a line-level error, the counting operation is allowed to continue, and if 10 *"BOUNCE_X@Y"* events have been observed in the window of 6 hours, output event *"X@Y unstable"* is produced.

```
# If a router interface is in down state for less
# than 15 seconds, generate event
# "<router> INTERFACE <interface> SHORT OUTAGE";
# otherwise generate event
# "<router> INTERFACE <interface> DOWN".

type=PairWithWindow
ptype=RegExp
pattern=(\S+) \d+: %LINK-3-UPDOWN: Interface (.+), changed
  state to down
desc=$1 INTERFACE $2 DOWN
action=event %s
ptype2=RegExp
pattern2=($1) \d+: %LINK-3-UPDOWN: Interface ($2), changed
  state to up
desc2=$1 INTERFACE $2 SHORT OUTAGE
action2=event %s
window=15

# If "<router> INTERFACE <interface> DOWN" event is
# received from the previous rule, send a notification and
# wait for "interface up" event for the next 24 hours.

type=Pair
ptype=RegExp
pattern=(\S+) INTERFACE (\S+) DOWN
desc=$1 interface $2 is down
action=shellcmd notify.sh "%s"
ptype2=RegExp
pattern2=($1) \d+: %LINK-3-UPDOWN: Interface ($2), changed
  state to up
desc2=$1 interface $2 is up
action2=shellcmd notify.sh "%s"
window=86400

# If ten "short outage" events from the first rule have been
# observed in the window of 6 hours, send a notification.

type=SingleWithThreshold
ptype=RegExp
pattern=(\S+) INTERFACE (\S+) SHORT OUTAGE
desc=Interface $2 at node $1 is unstable
action=shellcmd notify.sh "%s"
window=21600
thresh=10
```

Figure 1. An example ruleset for Cisco router *syslog*-messages.

III. SEC Performance And Application Experience

SEC has modest memory and CPU time requirements, and can therefore be installed even on older workstations (it has been successfully used on Linux nodes with Intel 80486 processors and 16MB of memory). Even when hundreds of event correlation operations and hundreds of contexts are simultaneously active and stored in the SEC working memory, the program consumes less than 5MB of memory on most architectures (depending on the OS platform, the code part takes about 3-4MB of it). In a recently conducted experiment, SEC was installed on a 1GHz PIII server, with the SEC rulebase containing 57 rules. The experiment lasted for about 17.5 days (1523599 seconds), with 47.5 input events arriving per second as an average. 9.2% of the 72390360 input events were found matching and were correlated. During the experiment, SEC consumed 4.7% of the CPU time.

The author has received feedback from more than 30 companies and individuals. Table I presents the results of a recent survey among some companies that are employing SEC.

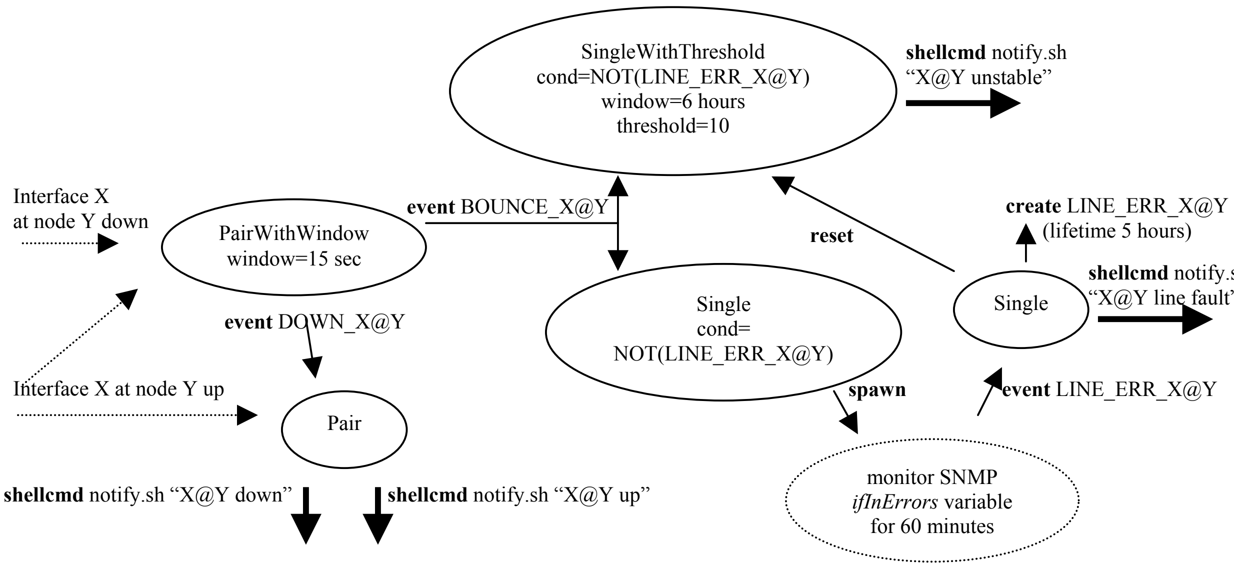
SingleWithThreshold
cond=NOT(LINE_ERR_X@Y)
window=6 hours
threshold=10
shellcmd notify.sh "X@Y unstable"
Interface X at node Y down
PairWithWindow
window=15 sec
event BOUNCE_X@Y
reset
create LINE_ERR_X@Y (lifetime 5 hours)
shellcmd notify.sh "X@Y line fault"
Single
Single
cond=
NOT(LINE_ERR_X@Y)
event DOWN_X@Y
Interface X at node Y up
Pair
spawn
event LINE_ERR_X@Y
monitor SNMP *ifInErrors* variable for 60 minutes
shellcmd notify.sh "X@Y down"
shellcmd notify.sh "X@Y up"

Figure 2. The work of a sample SEC ruleset (input events are depicted as dashed arrows and output events as bold arrows; circles represent event correlation operations and the dashed circle represents an external fault analysis module; regular arrows represent SEC actions).

TABLE I. THE RESULTS OF THE SEC USER SURVEY.

Type of the company	**Location**	**Description of the managed network**	**How SEC is applied**	**Advantages of SEC over other event correlation systems**
Banking card authorization center	Europe	30 servers, routers, and firewalls	Event correlation engine for NMS and IDS, logfile monitoring and system monitoring. An important application of SEC is fraud detection.	Straightforward, easy, and transparent configuration and rule definition system.
Technology-based marketing agency	US	600 nodes across US and UK	Gather and correlate service issues from Cisco CSS content switches.	Power and control in the amount you choose.
Financial institution	US	6000 workstations, 400 servers, 350 switches, 250 routers (distributed over US plus 5 other countries)	Used as a central event correlation engine for HP OpenView NNM. Also used for central monitoring of *syslog*-messages from Cisco devices.	More flexible and customizable than other event correlation systems.
Retail sales of consumer electronics	US	8000 managed nodes; the company WAN covers continental US, Alaska, Hawaii, and US territories	Network management with HP OpenView, logfile monitoring, dynamic webpage generation, etc.	SEC provides a low cost and efficient method to plug in event correlation and event management into HP OpenView.
Telecommuni-cations Carrier/ Provider	US	One of the largest international networks in the world	Logfile monitoring (collect and interpret alarms at call centers, and send a notification to a national support team).	Good level of control over monitoring triggers.
Network consulting	Global (more than 30 offices in US, Europe, and Asia)	SEC is used in the US network of a major European car manu-facturer (100 routers, 300 switches)	Used as a correlation engine for Cisco DFM platform and for Snort IDS.	Provides event correlation without significant programming resources, runs on multiple platforms, integrates well with external scripting languages.
Software development, IT consulting and services	Global (offices in Europe, US, Asia, Australia, South-America)	Global network, spread worldwide across the globe	Used as a prototype for event correlation experiments.	Free download status.

SEC has been used on a variety of OS platforms, like Linux, Solaris, HP-UX, AIX, FreeBSD, Tru64 UNIX, and Windows2000. It has been applied in various domains, like network fault and performance management, intrusion detection, logfile monitoring and analysis, and fraud detection. It has also been used for academic research [17]. Applications that SEC has been integrated with include HP OpenView Network Node Manager, HP OpenView ITO (both management server and agents), CiscoWorks, and Snort IDS. Equipment managed by SEC ranges from servers, routers, and switches to cable modem devices.

IV. Availability Information

SEC was released on March 23, 2001, and is distributed under the terms of GNU General Public License. Its main download page is located at *http://kodu.neti.ee/~risto/sec/*, and since March 2002 a mirror page at *http://simple-evcorr.sourceforge.net* is also available.

The SEC users mailing list can be joined at *https://lists.sourceforge.net/lists/listinfo/simple-evcorr-users/*.

V. Future Work

For a future work, the author plans to create a public rule library for applications and devices managed with SEC (i.e., standard rulesets for Snort, Cisco devices, etc.), and to include support for distributed event correlation in SEC.

Also, the author plans to extend SEC with fault analysis modules that employ various AI techniques, and to use SEC in experiments together with expert system shells and data mining tools.

Acknowledgments

The author thanks all the people who have provided feedback about SEC, and all the companies that participated in the SEC user survey.

The author expresses his gratitude to Al Sorrell, James Brown, John P. Rouillard, Jon Frazier, Mark D. Nagel, and Rick Casey, who have supplied software patches and documentation updates.

Also, the author wishes to thank CIO Mr. Tõnu Liik, CTO Dr. Paul Leis, and DSO Mr. Kaido Raiend from the Union Bank of Estonia for their kind support.

References

[1] G. Jakobson, M. Weissman, L. Brenner, C. Lafond, C. Matheus, “GRACE: Building Next Generation Event Correlation Services”, *Proceedings of the 7th Network Operations and Management Symposium*, pp. 701-714, April 2000.

[2] Stuart Staniford, James A. Hoagland, Joseph A. McAlerney, “Practical Automated Detection of Stealthy Portscans”, in press (to appear in the *Journal of Computer Security*).

[3] S. Staniford-Chen et al., “GrIDS – A Graph-Based Intrusion Detection System for Large Networks”, *Proceedings of the 19th National Information Systems Security Conference*, pp. 361-370, October 1996.

[4] Martin Roesch, “Snort – Lightweight Intrusion Detection for Networks”, *Proceedings of USENIX 13th System Administration Conference*, pp. 229-238, November 1999.

[5] Stephen E. Hansen and E. Todd Atkins, “Automated System Monitoring and Notification With Swatch”, *Proceedings of USENIX 7th System Administration Conference*, pp. 145-152, November 1993.

[6] Wolfgang Ley and Uwe Ellerman, logsurfer(1) and logsurfer.conf(4) manual pages, unpublished (see *http://www.cert.dfn.de/eng/logsurf/*).

[7] Matt Bing and Carl Erickson, “Extending UNIX System Logging with SHARP”, *Proceedings of USENIX 14th System Administration Conference*, pp. 101-108, December 2000.

[8] Hewlett-Packard Company, *Event Correlation Services – Designer’s Guide*, HP document J1095-90304, 1998.

[9] SMARTS, *http://www.smarts.com*.

[10] NerveCenter, *http://www.open.com/htm/nervecenter.htm*

[11] Shane O’Donnell, “Network Management: Open Source Solutions to Proprietary Problems”, *Proceedings of the 28th SIGUCCS Conference on User Services*, pp. 208-217, October 2000.

[12] OpenNMS, *http://www.opennms.org*.

[13] CLIPS, *http://www.ghg.net/clips/CLIPS.html*.

[14] G. Jakobson and M. Weissman, “Real-time telecommunication network management: Extending event correlation with temporal constraints”, *Proceedings of the 4th International Symposium on Integrated Network Management*, pp. 290-301, May 1995.

[15] Risto Vaarandi, “Platform Independent Event Correlation Tool for Network Management”, *Proceedings of the 8th Network Operations and Management Symposium*, pp. 907-909, April 2002.

[16] Peter Wainwright et al., *Professional Perl Programming*, Birmingham, UK, Wrox Press Ltd., 2001.

[17] Hugh R. Casey, *The Simple Event Monitor, A Tool for Network Management*, MSc thesis, University of Colorado, 2002.