Towards a Grid-wide Intrusion Detection System

Stuart Kenny and Brian Coghlan

Trinity College Dublin, Ireland
{stuart.kenny, coghlan}@cs.tcd.ie

Abstract. We describe SANTA-G (Grid-enabled System Area Networks Trace Analysis), an instrument monitoring framework that uses the R-GMA (Relational Grid Monitoring Architecture). We describe the CanonicalProducer, the component that allows for instrument monitoring, and how it would be used to construct the basis of a Grid-wide intrusion detection system.

1 The R-GMA

The Grid Monitoring Architecture (GMA) [2] of the Global Grid Forum (GGF), as shown in Figure 1, consists of three components: *Consumers*, *Producers* and a directory service, which in the R-GMA is referred to as a *Registry*.

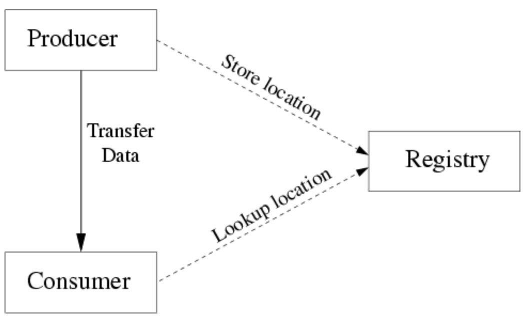

Fig. 1. Grid Monitoring Architecture

R-GMA is a relational implementation of the GMA developed within the European DataGrid (EDG), which harnesses the power and flexibility of the relational model. R-GMA creates the impression that you have one RDBMS per Virtual Organisation (VO). However it is important to appreciate that the system is a way of using the relational model in a Grid environment and *not* a general distributed RDBMS with guaranteed ACID properties. All the producers of information are quite independent. It is relational in the sense that Producers announce what they have to publish via an SQL CREATE TABLE

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 275–284, 2005.

statement and publish with an SQL INSERT and that Consumers use an SQL SELECT to collect the information they need. For a more formal description of R-GMA see [3]. The R-GMA makes use of the Tomcat Servlet container. Most of the R-GMA code is written in Java and is therefore highly portable. The only dependency on other EDG software components is in the security area.

There have so far been defined not just a single Producer but four different types: a DataBaseProducer, a StreamProducer, a LatestProducer and a CanonicalProducer. All appear to be Producers as seen by a Consumer, but they have different characteristics. The StreamProducer allows for information to be streamed continously to a Consumer. The LatestProducer only stores the most recent tuple of information for a given primary key, thus providing the latest-state information, whereas a DataBaseProducer stores the entire history of a stream of information.

2 The CanonicalProducer

If we have to deal with a large volume of data it may not be practical to convert it all to a tabular storage model. Moreover, it may be inefficient to transfer the data to a Producer servlet with SQL INSERT statements. It may be judged better to leave the data in its raw form at the location where it was created. The CanonicalProducer is able to cope with this by accepting SQL queries and using user-supplied code to return selected information in tabular form when required.

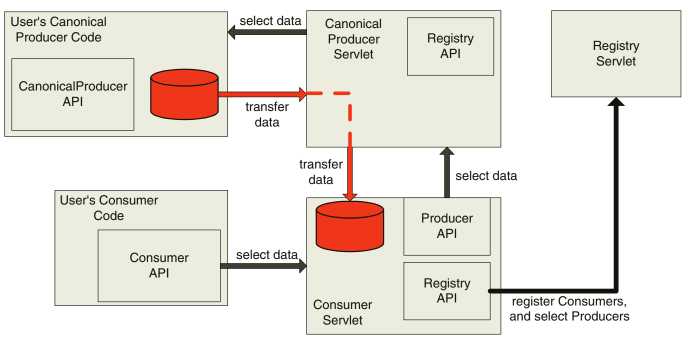

Fig. 2. CanonicalProducer Servlet Communication

In general the R-GMA producers are sub-classes of the Insertable class, the class that provides the insert method. The insert method is used by the producers to send data to the servlets as an SQL INSERT string. The CanonicalProducer is different however; it is a subclass of the Declarable class. This means that it inherits the methods for declaring tables, but not inserting data. The user's producer code is responsible for obtaining the data requested. Figure 2 shows the communication between the servlets for a CanonicalProducer. When the other producer types publish data, the data is transferred to a local producer servlet via a SQL INSERT. The CanonicalProducer Servlet, however, is never sent raw data, which is instead retained local to the user's CanonicalProducer code.

Because the user must write the code to parse and execute the query, the CanonicalProducer can be used to carry out any type of query on any type of data source.

R-GMA is being further developed within the EU EGEE project [23]. A new Static query type has been added. The concept of the Canonical Producer has been extended as an On-Demand Producer that can encompass large databases as well as instruments, and which specifically supports the static query type.

3 SANTA-G

SANTA-G (Grid-enabled System Area Networks Trace Analysis) is a generic template for ad-hoc, non-invasive monitoring with external instruments, see Figure 3.

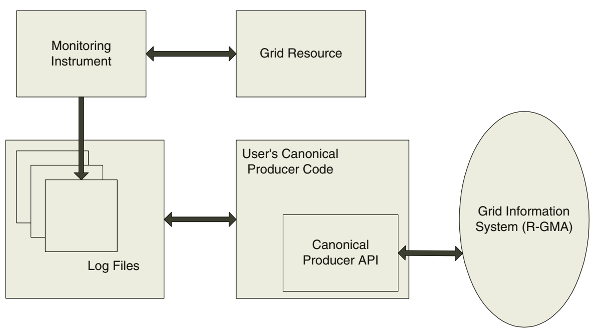

Fig. 3. SANTA-G monitoring framework

The template allows for the information captured by external instruments to be introduced into the Grid Information System. It is possible for these intruments to be anything, from fish sonars to PCR Analysers. The enabling technology for the template is the CanonicalProducer. The demonstrator of this concept, developed within the CrossGrid [17] project, is a network monitor that allows a user to access logs stored in *libpcap* (a packet capture library) format *through* the R-GMA. Examples of tools that generate logs in this format are Tcpdump [13], an open-source packet capture application, and SNORT.

SNORT [14] is an open-source network intrusion detection system. SNORT works by monitoring network packets received on the host's network interface card. Any packet found which matches a rule from a set of defined security rules is logged to a log file, and an alert is generated, which is also stored in a separate alert file.

4 Grid-wide Intrusion Detection

We now describe the use of the SANTA-G with SNORT as the basis for Grid-wide intrusion detection.

The SANTA-G NetTracer is composed of three components that allow for the monitoring data to be accessed through the R-GMA: a Sensor (which is installed on the node(s) to be monitored), a QueryEngine, and a Viewer GUI (see Figure 4). The Sensor monitors the log files created by the external sensor, for example SNORT, and notifies the QueryEngine when new log files are detected. SNORT logs alerts when suspect packets are detected. The Sensor monitors the alert file generated by SNORT and when a new alert is detected its details are sent to the QueryEngine, which records these events and publishes them to users through the R-GMA (by using the LatestProducer API). Users can then view these alerts, using the Viewer GUI. The full packet data of the packet that triggered the alert can also then be viewed by querying the packet log file also generated by SNORT.

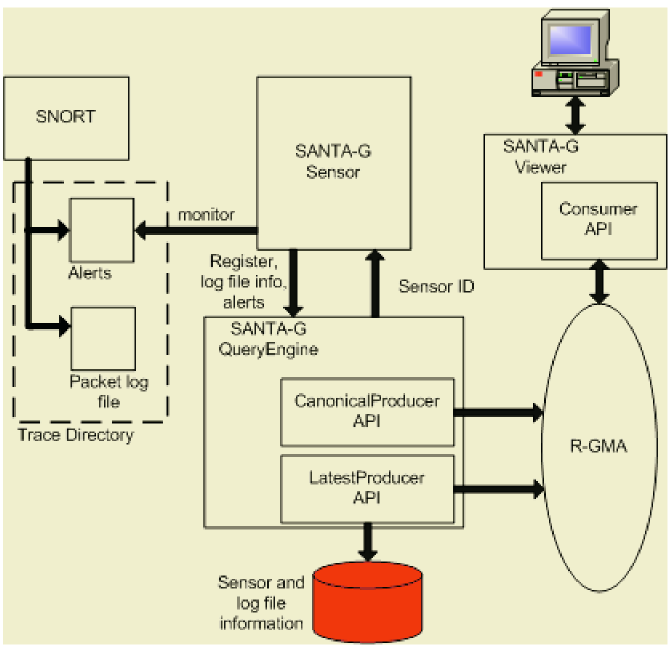

Fig. 4. SANTA-G NetTracer, SNORT monitoring

The QueryEngine provides the interface to the R-GMA by using the CanonicalProducer API. Data is viewed via the R-GMA by submitting an SQL SELECT statement, as if querying a relational database. Through the CanonicalProducer this query is forwarded to the QueryEngine, which then parses the query, searches the appropriate log file to obtain the data required to satisfy the query, and returns the dataset to the GUI through the R-GMA.

The QueryEngine implements the required components of CanonicalProducer code. Figure 5 shows how the QueryEngine executes a SQL query received from the R-GMA (i.e. from the CanonicalProducerServlet).

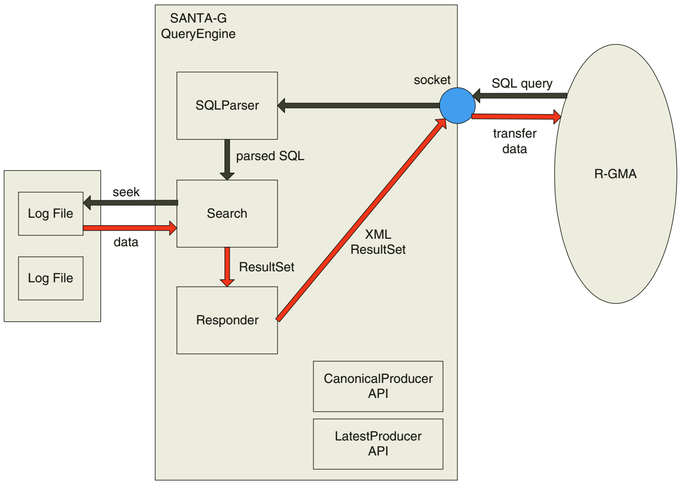

Fig. 5. SANTA-G QueryEngine Query Processing

The QueryEngine listens on a socket, waiting for connections from the Servlet. When a connection is made the SQL query is read from the socket and passed to an SQLParser class. The parser breaks the query into three separate lists; a select list that contains the fields to be read, a from list that contains the table the fields belong to, and a where list that contains the values used to filter the fields with. The Search class searches the log file for data that matches the WHERE predicates specified in the query, and extracts the required fields. The data that satisfies the query is accumulated into a ResultSet in XML format and returned to the Servlet over the socket connection. For example, the following query:

```
SELECT source_address, destination_address,
packet_type
FROM Ethernet
WHERE sensorId = 'some.machine.com:0'
AND fileId = 0
AND packetId < 100
```

would return the source address, destination address, and packet type fields of the Ethernet header for the first 100 packets in the log file assigned ID 0 and stored on 'some.machine.com'.

The information that can be queried is defined by the schema that the CanonicalProducer registers with the CanonicalProducer servlet. The schema for the SNORT alerts is shown in Table 1.

Table 1. SNORT alerts table schema

Field	**Key**	**Description**
siteId	PRI	Site ID
sensorId	PRI	Sensor ID
fileId	PRI	Log file ID
alert_timestamp		Timestamp of when the event was logged
alert_type		Type of event
sig_info		Alert signature information
message		Alert message
classification		Alert classification name
priority		Alert priority
source_ip		Source IP address
destination_ip		Destination IP address
source_port		TCP source port
destination_port		TCP destination port
data		Packet header data

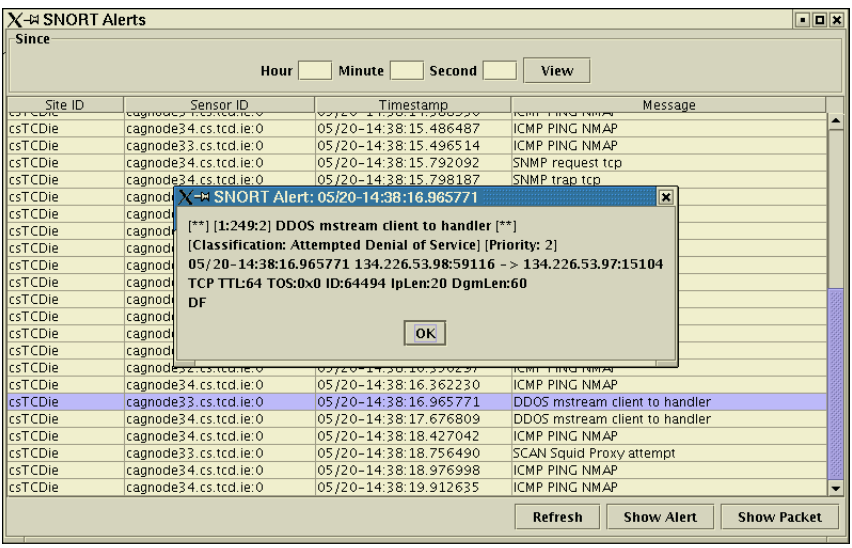

Fig. 6. SANTA-G Viewer, SNORT alerts panel

The SANTA-G Viewer provides a graphical user interface, which makes use of the R-GMA Consumer API, to allow users to graphically view network packets in the log files, and also to build and submit SQL queries that will be carried out on the log files. Figure 6 shows the SNORT alerts panel of the Viewer, which allows a user to browse the alerts that have been published by the SNORT sensors to the R-GMA.

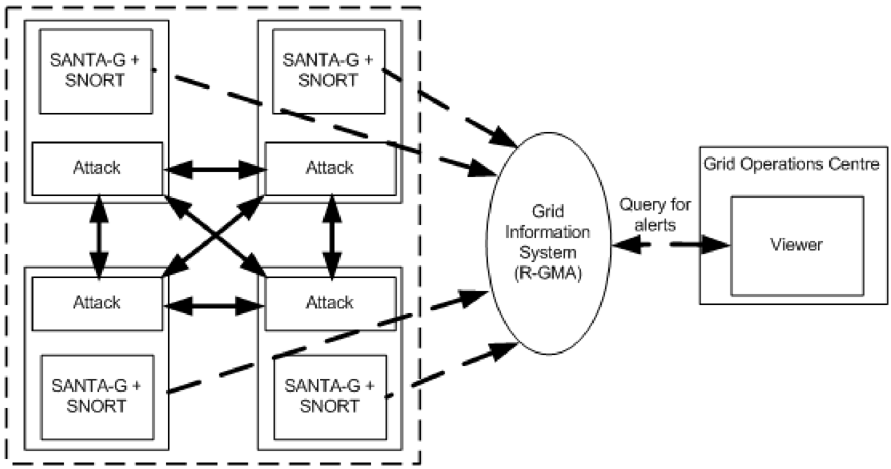

Fig. 7. SNORT monitoring example

Figure 7 shows how alerts can be generated, and published from a site, in order for them to be viewed at a Grid Operations Center. Here we show four worker nodes instrumented with the SNORT sensors, each attacking the other three. The alerts generated will be collected and published to the R-GMA via the QueryEngine. The Viewer (or any R-GMA Consumer) can then be used to query for and view the published alerts. If this example is expanded to include multiple sites then the alerts table published by the R-GMA becomes a Grid-wide intrusion log.

5 Future Work

It is intended to use the SNORT functionality of the SANTA-G network monitoring tool developed within Grid-Ireland to construct a 'Grid-wide intrusion detection system'. It is envisaged that each site within a Grid will run a set of SNORT sensors publishing alerts to the R-GMA. A high-level incident detection, tracking and response platform will be created by using custom coded Consumers to filter and analyse the alerts published in order to detect patterns that would signify an attempted distributed attack on the Grid infrastructure. This will be constructed with two components (as shown in Figure 8):

1. a R-GMA Archiver that will query for the alerts detected by the SNORT sensors and save the results in a MySQL database, that will then represent the Grid-wide intrusion log, and

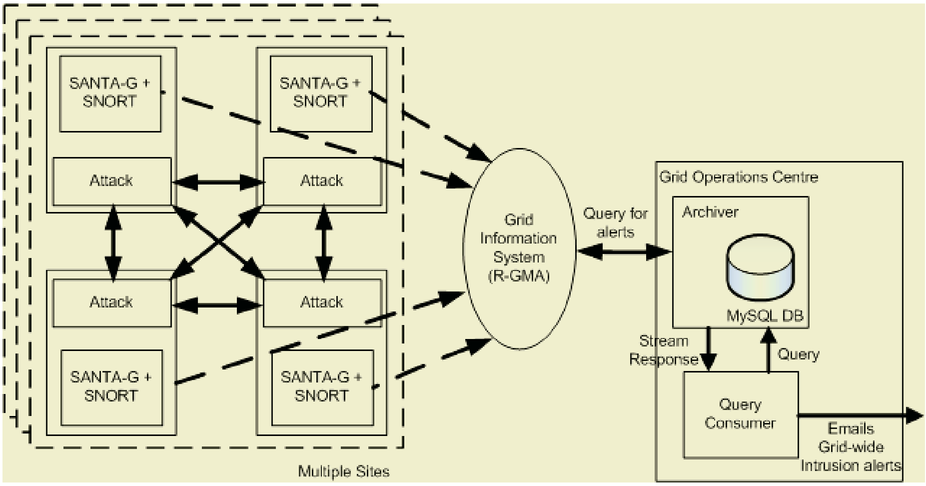

Fig. 8. Grid-wide Intrusion Detection System

2. one or more R-GMA Consumers that will issue stream queries to the Archiver, and receive back a stream of responses that they will convert to email alerts.

It is also intended to complement the SNORT data with information from other security components. Work is currently underway creating new sensor types and query engines for use with such tools as Tripwire [15] and AIDE (Advanced Intrusion Detection Environment) [16].

6 Conclusion

The SANTA-G network monitoring tool developed within the CrossGrid project demonstrates the CanonicalProducer concept by allowing log files to be accessed *through* the R-GMA. Since the logs are published to the R-GMA grid information system this can provide the basis for a Grid-wide intrusion detection system. SANTA-G NetTracer has been extended to publish logs generated by the SNORT network intrusion detection system. We intend to use this functionality, along with further extensions that incorporate information from other security components, to construct a system that will allow for Grid-wide intrusion detection. By using other R-GMA components, such as an Archiver, alerts published from multiple sites can be aggregated to form a Grid-wide intrusion log. Custom coded Consumers can then query this log for specific alert patterns, triggering their own alerts if a pattern is detected, and thereby generating Grid-wide intrusion alerts.

References

1. Andrew Cooke, Werner Nutt, James Magowan, Manfred Oevers, Paul Taylor, Ari Datta, Roney Cordenonsi, Rob Byrom, Laurence Field, Steve Hicks, Manish Soni, Antony Wilson, Xiaomei Zhu, Linda Cornwall, Abdeslem Djaoui, Steve Fisher, Norbert Podhorszki, Brian Coghlan, Stuart Kenny David O'Callaghan, John Ryan. RGMA: First Results After Deployment CHEP03, La Jolla, California, March 24-28, 2003.
2. B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, and R. Wolski. *A Grid monitoring architecture.* Global Grid Forum Performance Working Group, March 2000. Revised January 2002.
3. Andy Cooke, Alasdair J G Gray, Lisha Ma, Werner Nutt, James Magowan, Manfred Oevers, Paul Taylor, Rob Byrom, Laurence Field, Steve Hicks, Jason Leake, Manish Soni, Antony Wilson, Roney Cordenonsi, Linda Cornwall, Abdeslem Djaoui, Steve Fisher, Norbert Podhorszki, Brian Coghlan Stuart Kenny, David O'Callaghan. *R-GMA: An Information Integration System for Grid Monitoring* Proceedings of the Tenth International Conference on Cooperative Information Systems, 2003.
4. Andrew Cooke, Werner Nutt, James Magowan, Manfred Oevers, Paul Taylor, Ari Datta, Roney Cordenonsi, Rob Byrom, Laurence Field, Steve Hicks, Manish Soni, Antony Wilson, Xiaomei Zhu, Linda Cornwall, Abdeslem Djaoui, Steve Fisher, Norbert Podhorszki, Brian Coghlan, Stuart Kenny, Oliver Lyttleton, David O'Callaghan, John Ryan. *Information and Monitoring Services Within a Grid* Proceedings of the Eleventh International Conference on Cooperative Information Systems, 2004.
5. Brian Coghlan, Abdeslem Djaoui, Steve Fisher, James Magowan, Manfred Oevers. *Time, Information Services and the Grid* 31st May 2001.
6. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, P. Vanderbilt. *Grid Service Specification* http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-04_2002-10-04.pdf, 2003.
7. The DataGrid Project. http://www.eu-datagrid.org
8. DataGrid WP3. *DataGrid Information and Monitoring Final Evaluation Report* https://edms.cern.ch/document/410810/4/DataGrid-03-D3.6-410810-4-0.pdf
9. Brian Coghlan, Stuart Kenny. *SANTA-G Software Design Document*
10. Brian Coghlan, Stuart Kenny. *SANTA-G First prototype Description* http://www-eu-crossgrid.org/Deliverables/M12pdf/CG3.3.2-TCD-D3.3-v1.1-SANTAG.pdf
11. CrossGrid WP3. *Deliverable D3.5, Report on the Results of the 2nd and 3rd Prototype* http://www-eu-crossgrid.org/Deliverables/M24pdf/CG3.0-D3.5-v1.2-PSNC010-Proto2Status.pdf
12. Brian Coghlan, Andrew Cooke, Roney Cordenonsi, Linda Cornwall, Ari Datta, Abdeslem Djaoui, Laurence Field, Steve Fisher, Steve Hicks, Stuart Kenny, James Magowan, Werner Nutt, David O'Callaghan, Manfred Oevers, Norbert Podhorszki, John Ryan, Manish Soni, Paul Taylor, Antony Wilson Xiaomei Zhu. *The CanonicalProducer: an instrument monitoring component of the Relational Grid Monitoring Architecture* Proceedings of the 3rd International Symposium on Parallel and Distributed Computing, July 2004.
13. Tcpdump. http://www.tcpdump.org
14. SNORT. http://www.snort.org
15. Tripwire. http://www.tripwire.org
16. AIDE. http://sourceforge.net/projects/aide

17. The CrossGrid Project http://www.eu-crossgrid.org
18. DataGrid WP3 Information and Monitoring Services http://hepunx.rl.ac.uk/edg/wp3/
19. Global grid forum. http://www.ggf.org
20. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*, chapter 2: Computational Grids, pages 15–51. Morgan Kaufmann, 1999.
21. I. Foster, C. Kesselman, and S. Tuecke. *The anatomy of the Grid: Enabling scalable virtual organization.* The International Journal of High Performance Computing Applications, 15(3):200–222, 2001.
22. Globus Toolkit. http://www.globus.org
23. Enabling Grids for E-science in Europe http://egee-intranet.web.cern.ch/egee-intranet/gateway.html