Information Assurance Intrusion Detection Sensor Database Design: Lessons Learned

Authors: Brian T. Spink,
Thomas M. Blake
AFRL/IFGB
525 Brooks Road
Rome, NY 13441
315-330-7596
315-330-1482

Vincent Salerno
LittonPRC, Inc
525 Brooks Road
Rome, NY 13441
315-33-3775

Emails: spinkb@rl.af.mil; blaket@rl.af.mil; salernov@rl.af.mil

Abstract:
Current architectural trends in information assurance for the DOD focuses on the fusion and correlation of large volumes of data collected across several intrusion detection systems and boundary devices. To be optimally effective this data must support near-real time analysis for immediate situational awareness, and long term trending, to identify subtle anomalies and suspicious events that could lead to compromise or denial-of-service. The obvious benefits of using a relational data model and SQL to help solve this problem continue to be observed at Air Force Research Laboratory (AFRL). Prototype integration environments for information assurance, using Oracle, have now been used in operational demonstrations. This paper explores the design implications and some operational pitfalls encountered integrating the Relational Database Management System (RDBMS) concepts into these prototype environments.

Background:
This work is being performed as part of the AFRL Automated Intrusion Detection Environment (AIDE) Advanced Concept Technology Development (ACTD). The top-level objectives of this program are to create an architecture for the sharing, integration, analysis and warning of IW attacks; incorporate current and maturing intrusion sensing tools in conjunction with expert systems technology for the management of distributed systems; and correlate intrusion events at local agency, Commander In Charge (CINC), and Joint command levels to tighten the detection grid and increase the success of identifying Information Warfare (IW) threats. The AIDE architecture has been evolving to a database-centric application. The database design is key in the successful implementation of AIDE. The database specific technical challenges being solved include:

- Data Normalization - Deconflict and store raw, multi-sensor data in common database(s). This is a challenging problem for AIDE with a large number of sensors generating a high volume of disparate sensor data.
- Correlation & Analysis - Correlate data from multiple sensors and support near-term and longer-term analysis of collected data.
- Automated Reporting - Near real-time, secure analyst reporting to decision-makers, intelligence and law enforcement communities.

DATABASE

Normalized Signature Table

The problem with data retrieval from multiple sensors includes the differences in their event signatures. AIDE normalizes the event signatures and ties the sensor's signature with the AIDE normalized signature. When an event is displayed it includes the AIDE normalized signature. The database web browser contains

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 01-01-2001	2. REPORT TYPE Conference Proceedings	3. DATES COVERED (FROM - TO) xx-xx-2000 to xx-xx-2000

4. TITLE AND SUBTITLE Information Assurance Intrusion Detection Sensor Database Design: Lessons Learned Unclassified	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Spink, Brian T. ; Salerno, Vincent ; Blake, Thomas M. ;	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME AND ADDRESS AFRL/IFGB 525 Brooks Road Rome, NY13441	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS Director, CECOM RDEC Night Vision and Electronic Sensors Directorate Security Team 10221 Burbeck Road Ft. Belvoir, VA22060-5806	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES
See Also ADM201258, 2000 MSS Proceedings on CD-ROM, January 2001.

14. ABSTRACT
Current architectural trends in information assurance for the DOD focuses on the fusion and correlation of large volumes of data collected across several intrusion detection systems and boundary devices. To be optimally effective this data must support near-real time analysis for immediate situational awareness, and long term trending, to identify subtle anomalies and suspicious events that could lead to compromise or denial-of-service. The obvious benefits of using a relational data model and SQL to help solve this problem continue to be observed at Air Force Research Laboratory (AFRL). Prototype integration environments for information assurance, using Oracle, have now been used in operational demonstrations. This paper explores the design implications and some operational pitfalls encountered integrating the Relational Database Management System (RDBMS) concepts into these prototype environments.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Public Release	18. NUMBER OF PAGES 7	19. NAME OF RESPONSIBLE PERSON Fenster, Lynn lfenster@dtic.mil
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39.18

this normalized table data showing what the sensor named the signature and an explanation of the detected signature. New signatures can be added as sensors discover more exploits and incorporate them into their product. This keeps AIDE current with the sensor's database and allows a user to edit the information as needed.

Oracle Database

The AIDE architecture is centered on an ORACLE 8.1.5 server. An AIDE installation is comprised of tables and entity relationships which were designed to reside in one of five categories: 1) Configuration, 2) Static, 3) Activity, 4) Anomalies, and 5) Summary information. Intrusion detection tools populate AIDE tables via the various bridges. These bridges written in PRO*C are designed to efficiently insert event type data into the database. ORACLE stored procedures and triggers are utilized to effectively manipulate and summarize data. Procedures such as: 1) Signature normalization, 2) PIPES, to the AIDE correlation engine are just a few of these procedures. ORACLE advanced queuing techniques are used to communicate events from one AIDE site to another. ORACLE tuning techniques have been initiated to increase the number of events inserted into the dB to approximately 200 records/sec. The AIDE GUI and WEB front end are then used to display various views of the data to the user.

Oracle Web Browser

AIDE utilizes ORACLE Application Server (OAS) 4.0.7. The Web front end is used to configure various tables needed by AIDE to perform such tasks as: 1) Communicate with sensors, 2) Communicate AIDE to AIDE, and 3) Data normalization. Views are also supplied which allow the user to query across reported sensor data searching for specific items of interest (i.e. Source IP's, Destination ports….). The WEB front end is a very powerful tool, which allows a user to drill down on defined set of data. The WEB browser reporting specific data perfectly compliments the AIDE near real time GUI, which reports current events for a given period of time. The two give the user a complete and detailed look into the data collected or generated by the sensors and AIDE correlation.

Advanced Queuing

In keeping with the AIDE database centric development philosophy, the AIDE system handles all communication between AIDE nodes through the database, utilizing the Advanced Queuing built in functionality. AQ is a guaranteed delivery queue based messaging system, which can be used for communication between Oracle databases as well as for inter-process communication on a single database instance. Messages contain an arbitrary data object and a set of addresses. To send a message, it is placed on a message queue.

Oracle takes care of routing the message to the correct destination queue if necessary. Once queued and routed, messages can be unqueued by a client or group of clients. Messages not deliverable, or which have expired, are moved onto an exception queue rather than being lost, so complete accountability of the messages is provided. Oracle manages all of the routing, error handling, and delivery functionality, making the AQ system very user friendly, as well as cross platform.

In AIDE, insertion or modification of an event in the event table fires a trigger. This automatically encapsulates the row into an object and places it on a transmission queue addressed to any remote sites that have been designated for automatic forwarding. Similarly, incoming messages are pulled off of the receive queue and placed into the event table. This insert may in turn cause the message to be retransmitted up the hierarchy (infinite loops are detected and removed before insertion). Events may also be transmitted manually to arbitrary destinations at the users request.

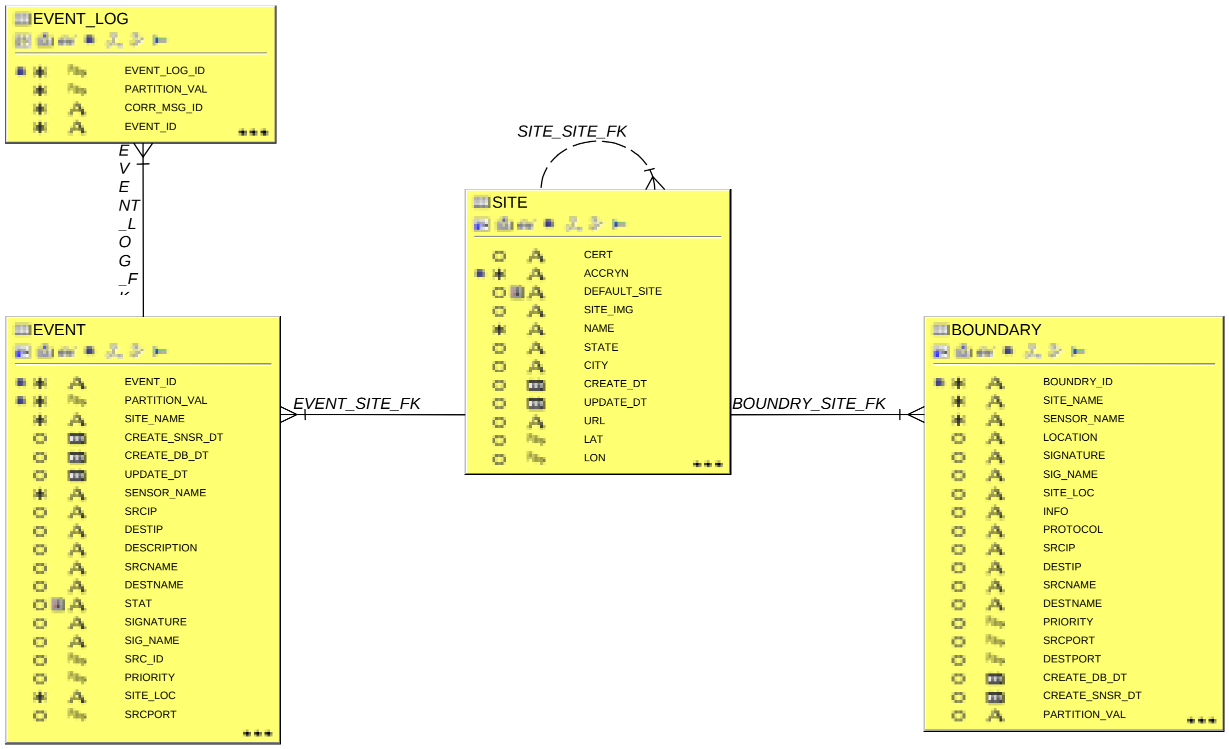
Using this strategy, we have been able to keep all messaging related functionality within the database, and have completely avoided the needed to write specialized networking code. Error handling is available if it becomes desirable to implement retransmission of events to sites, which were unreachable for some period. In addition, we can, if desired, maintain a complete log of all communications activity.

Data Base Schema:
The entity relationships and table structure used was designed to reside in five categories:
1- Configuration, 2 -Traffic, 3- Static, 4- Anomalies, and 5- Historic. For the sake of clarity a subset of the tables have been broken down into their categories.

Category	**Table**
Configuration	HOST
	HOST_SERVICE
	SENSOR_SIGNATURE
	AIDE_SIGNATURE
	SIGNATURE_CATEGORY
	CORR_CATEGORY
	COMM_STATUS
Traffic	EVENT
	BOUNDARY
	HOST_EVENT
	CORRELATED_EVENT
	EVENT_LOG
Static	SERVICE
	SITE
	SENSOR
Anomalies	BAD_IP
	INCIDENTS
Historic	ARCHIVE
	EVENT_HIST
	EVENT_LOG_HIST

At the heart of the AIDE database is the traffic category. All intrusion detection sensors that interface with the AIDE system write data to one or more of these tables. Diagram 1 details the relationship between some of these tables.

Diagram 1: Traffic



The usage of these tables is driven in part by the security tools (called sensors) chosen for integration into the AIDE system. The traffic tables used so far are:

- EVENT: This table contains all network based intrusion detection tools data. (REAL SECURE, NETRANGER).
- BOUNDARY: Data collected from boundary type sensors (FIREWALLS, ROUTERS).
- HOST_EVENT: Data collects from host based intrusion detection tools.
- CORRELATED_EVENT: Data generated by AIDE's correlation engine.
- EVENT_LOG: Drill down information on correlated events.

Database Population:
Population of the database tables includes record inserts, updates and deletions. In the AIDE environment these functions are handled in one of three ways:

- Bridging
- SQL*Loader
- Internally Generated

Bridging:

Data from stand-alone security tools, referred to as "sensors", must pass into the AIDE environment for processing. Generally all data will pass into the dB where rules, procedures and functions can act upon the data. It became clear early in development that the main objective was to optimally tune the bridges and dB to handle the large amounts of traffic generated by Intrusion Detection (ID) tools.

The initial design for inserting data utilized ORACLE's OCI (Oracle Call Interface). OCI is a set of function calls and library routines allowing DML (Data Manipulation Language) activity from within third generation programming languages like C. This capability is provided with the Oracle license at no extra charge. Our current approach uses Pro*C ®. This is a layered product offered by Oracle at additional cost. It allows SQL statements to be embedded in native code. This code would then be processed through a pre-compiler yielding object code that accesses the database directly.

Another issue we struggled with was trying to increase the insertion rate into the dB. To be of any use the data we were loading via bridges had to be as real time as possible. Our initial approach was to insert one record at a time committing after X number of records. The number of inserts committed as a group yielded commit rates between 10-30 records per second. Our second and current approach utilized array inserts. This approach allows the bridge to collect data from the sensors and store the data in an array. When the array is full the bridge performs one insert into the dB, dramatically increasing performance. The trick here was to determine the optimal array size. This was accomplished by trial and error using a static set of data being processed through the bridge. This method currently yields commit rates in the neighborhood of 100-125 records per second.

SQL*Loader®:
This method was reserved primarily for population of Static and Configuration tables. Oracle's utility is versatile and well documented. This tool is used for insert (modes are append to current data or replace all contents) only. The major limitation is that there is a fair amount of manual work required. However, once completed, tables can be reloaded with relative ease.

Internally Generated:
This population method is accomplished in one of two ways:

- DML originated via triggers or embedded procedures.
- DML originated from bridges and correlation engines.

Oracle offers a 4GL called PL/SQL ®. AIDE uses PL/SQL in the form of triggers, functions and procedures. Once compiled and instantiated these routines allow action to be performed based upon specific events on various tables. Triggers "fire" or execute based on DML actions on specific tables. Currently all triggers react to INSERT operations on the traffic tables. Either before or after an insert, the data involved in the transaction can be analyzed and various actions taken based on finding. This is how AIDE Normalizes data captured by a variety of sensors. They also provide the rudiments of a correlation capability within the database itself. One example of this is the ability for the dB to downgrade/upgrade a priority of an event based upon findings in other tables. In general, triggers are small programs with very specific purposes. Procedures on the other hand are not event driven. They can be much larger and have much more capability built in.

As mentioned earlier, AIDE has the ability to perform correlation on the data being processed by bridges. These correlation engines also have the ability to perform DML operations on various objects in the dB. This mechanism is used to modify dB objects based upon findings computed internally by the correlation engine.

Normalization:
The problem with data retrieval from multiple sensors includes the differences in their event signatures. AIDE normalizes the event signatures and ties the sensor's signature with the AIDE normalized signature. When an event is displayed it includes the AIDE normalized signature. The database web browser contains this normalized table data showing what the sensor named the signature and an explanation of the detected signature. New signatures can be added as sensors discover more exploits and incorporate them into their product. This keeps AIDE current with the sensor's database and allows a user to edit the information as needed.

Operational Impacts:
As stated earlier, system goals include near real-time situation assessment and the ability to perform long-term trend analysis. An RDBMS is expected to help achieve these goals. Continuous traffic table inserts guarantee optimum correlation. If the ID sensor is reporting traffic, then the database must be available to store the information.

One of the major issues confronting the AIDE project was how to deal with the tremendous amount of data generated by multiple sensors. Several approaches were researched to determine the fastest way to export data without hindering performance.
These include:

- Table Locking

- Database Downtime
- Specialized scripts
- Table Partitioning

Table Locking:
Our first approach was to lock a given table, create a copy of it to be exported using the ORACLE export routine, then truncate the original table. This method was deemed impractical because of the continuous data feed from ID sensors. The table lock would also leave a gap in our data that was not desired.

Database Downtime:
The next approach involved bringing the dB down, thus suspending any writes from ID sensors and leaving undesired gaps. Once the dB was down, certain hot tables could be exported via the ORACLE export routine then truncated. It was found that the export/truncate worked very quickly and only suspended the dB for a short time. The problem arose when trying to import the data. At times it took ORACLE import several hours to retrieve one day's worth of data.

Specialized Scripts:
Another approach was centered on a PL/SQL® script, which duplicated a hot traffic table. A cursor was created from the duplicate table to query the original table for a matching key. This gave us the ability to keep the dB up and receiving new data while deleting those records that had already been copied to the duplicate table. The duplicate table was then used as the export table. This method spawned other problems related to the confirmation of the ORACLE rollback segments, which had to be resized to handle the deletions.

Table Partitioning:
With ORACLE 8.x the ability to split tables and indexes into smaller pieces, called subpartitions allowed us another means of meeting the demands of a very large database (VLDB). Each partition when created, has the same logical attributes. For example, all partitions (or subpartitions) in a table share the same column and constraint definitions, and all partitions in an index share the same index options.
This allowed us to:

- Backup and recover each partition independently
- Improve manageability, availability and performance

In other words, we could archive certain subpartitions of a table without effecting the partition that was currently being written to by ID sensors. Thus eliminating any downtime for scheduled maintenance of any partitioned table. It must be noted that ORACLE partitioning is an option and therefore an extra cost.

AIDE communications:
In keeping with the AIDE database centric development philosophy, the AIDE system handles all communication between AIDE nodes through the database, utilizing the Oracle's Advanced Queuing built in functionality. AQ is a guaranteed delivery queue based messaging system which can be used for communication between Oracle databases as well as for inter-process communication on a single database instance. Messages contain an arbitrary data object and a set of addresses. To send a message, it is placed on a message queue. Oracle takes care of routing it to the correct destination queue if necessary. Once queued and routed, messages can be dequeued by a client or group of clients. Messages not deliverable, or which have expired, are moved onto an exception queue rather than being lost, so complete accountability of the messages is provided. Oracle manages all of the routing, error handling, and delivery functionality, making the AQ system very user friendly, as well as cross platform.

In AIDE, insertion or modification of an event in the event table fires a trigger, which automatically encapsulates the row into an object and places it on a transmission queue addressed to any remote sites which have been designated for automatic forwarding. Similarly, incoming messages are pulled off of the receive queue and placed into the event table. This insert may in turn cause the message to be retransmitted up the hierarchy (infinite loops are detected and removed before insertion). Events may also be transmitted manually to arbitrary destinations at the user request.

Using this strategy, we have been able to keep all messaging related functionality within the database, and have completely avoided the need to write specialized networking code. Error handling is available if it becomes desirable to implement retransmission of events to sites, which were unreachable for some period. In addition, we can, if desired, maintain a complete log of all communications activity.

We are planning to expand our use of the AQ facility in the future in order to make delivery of events to the GUI clients more efficient. Currently the clients must poll the database periodically and do a select across the event table looking for new events. Under the new scheme, the GUI clients will be notified on arrival of an event.

CONCLUSION

AIDE is a unique framework that allows data feeds from multiple and cross-platform sensors to provide a comprehensive evaluation of network traffic. This multiple sensor feature allows a stereo view utilizing the best features of the different sensors. Some sensors detect attacks by utilizing a pattern-matching word by word detection and comparing the words to the signature database. This is used to see things like /etc/passwd, chmod, /gci/phf and hundreds of others. Other sensors utilize a signature-matching paradigm by breaking the packets down to see what the packets are supposed to do at the destination. These sensors detect signatures like IP half scan, Satan scans, and Port scans along with hundreds of other signatures. Firewall log data will display connection denied data to enable the operator determine if an attacker is trying to connect to their network and the firewall is blocking them. This could be an attempted information gathering or vulnerability probe of the network. By displaying the firewall log data along with IDS data files, AIDE enables the operator to determine if the firewall rule set is correctly configured. Firewall rule sets change as users need short and long term holes enabled for project support. The display of firewall log files ensures that the expired requests are removed from the rule set thereby plugging up the holes through the firewall.

AIDE provides a cyber situational awareness of the sites under protection. A large company will have multiple firewalls and sensors to help provide multiple layers of protection against intruders. The data from different sensors and firewalls needs to be centralized for a composite picture of the network performance and security. The network security operator needs the information deciphered and filtered to reduce the information overload that is inherent with the protection of high-speed networks. The AIDE system is an approach to manage the intrusion detection data being collected and to automate to a large extent the function of the network protection analyst.