

# Network Intrusion Detection System with Data Mart

R. A. Wasniowski

Computer Science Department  
California State University Dominguez Hills  
Carson CA, USA

**Abstract** - *Network Intrusion Detection Systems (NIDS) capture large amounts of data that is difficult or impractical to report and analyze directly from the capture device. It is also common to have more than one NIDS device and reporting from a consolidated multi-NIDS device. To provide a platform for multi-NIDS device reporting and analysis, this paper describes a consolidated database, or Data Mart design and implementation to store data from multiple Snort NIDS devices. This consolidated Snort Mart is designed for reporting and analysis and can provide a platform for better understanding of NIDS device information*

**Keywords:** intrusion detection, database, rules.

## 1 Introduction

Intrusions are rapidly becoming one of the most important threats for information systems. A major concern is the high rate of false alarms produced by systems designed to detect intrusions called Intrusion Detection Systems (IDS). Network intrusion detection is commonly thought of as the process of determining when unauthorized people are making an attempt to break into network. However, this is not a complete picture of network intrusion detection. Though unauthorized login attempts is an easy to understand example of an intrusion, there are other types of activity that are not as clear, such as probing network with port scans or pings. Though not a direct attempt to break into network, these types of activities are a typical precursor to more hostile activity, and thus are considered an intrusion and should be identified as such. In general, there are three categories of Intrusion Detection systems: host based, network based, and application based [2, 6]. This classification depends on the type of sensors they use to collect data in order to detect possible attacks. In the host based approach every host has its own IDS agent and it collects data by monitoring connection attempts to various ports. A network based IDS collects data at the network level. Their sensors are located somewhere in the network and monitor network traffic. The third type processes data from running applications as input. Intrusion Detection Systems are usually not

implemented by using just a single concept, but multiple concepts to gather information to detect anomalous behavior in the system. Many organizations will have more than a single system and/or networks. Intrusion detection monitoring of these multiple systems and networks requires the existence of multiple intrusion detection systems. This is because each intrusion detection system must be connected to the networks being monitored. Also, bandwidth limitations more than one intrusion detection system may be required on the same network.

One of the most troublesome issues that an intrusion detection system administrator faces is what is called the “false positive.” This is basically an event identified as an intrusion attempt, but in reality it is not. The typical response to this by the administrator is to reconfigure the intrusion detection system to not identify that particular event as an intrusion attempt. However, this means that a real intrusion attempt may be missed if the intrusion detection system is configured not to identify the event as such. On the other hand, being constantly notified by “false positives” may also result in a false sense of security, as the administrator can adopt an attitude that typically intrusion events reported by the intrusion detection system are false positives and may not properly respond to a real intrusion attempt. So to be able to have the opportunity to be able to respond to more intrusion events, the intrusion detection system must be configured in a way that will probably report many of these “false positives” and thus, will process and store more information that is typically expected. Along with this is a greater effort required to review this information – though the intrusion detection system can identify store and event notify an administrator of an intrusion event, a person must still access and review the data to analyze the event, and typically will need to be able to review this event with others as well. This is because an intrusion attempt may be targeted at multiple systems and or networks in parallel or in series within some type of measurable timeframe.

Additional tools are required to better understand all of the data generated by the intrusion detection systems. This

need is not new, there are many existing solutions that will read from and produce reports from intrusion detection system logs. However, most of these tools are designed to directly read from the intrusion detection system log and are not optimized for reporting. Because of this, creating custom reports for analysis is more difficult and will often take a lot of time to produce information for analysis. This can result in these types of tools not being used, as they are too difficult and frustrating to use on a regular basis.

The purpose of this paper is to discuss implementation of prototype multi sensor based intrusion detection system. We are especially interested in analyzing traffic that has an abnormal or malicious character and should prompt a closer look. In this paper we propose a framework for multi sensor based intrusion detection system to support such analyzes and response. A specific feature of the model is that the systems use multiple sensors and mining to process log files. This reduces the overhead in a distributed intrusion detection system. We have developed a prototype based on this framework.

## 2 Background

There are several intrusion detection systems, and one of the most popular in public domain is Snort. Snort is an open source network packet monitoring and Intrusion Detection System. Snort looks for attack signatures, which are specific patterns of activity that has been defined to be of a suspicious or malicious intent. Snort analyzes network packets, and thus is classified as a Network Intrusion Detection System. These types of systems must be connected to the networks that they monitor and unless the network topology is very simple, multiple Snort systems, called Snort sensors, must be setup and configured to monitor these networks. Snort relies on the ability to recognize attack signatures in order to identify an attack. These pattern recognition definitions are called rules. Attacks are not static, as they are continuously evolving as systems are protected to withstand existing attack methodologies. Thus, it is critical to perform analysis of prior activity to look for trends or changes in activity that are not typically classified as an attack which are often the precursor to an attack.

As indicated by Cox and Gerg [7], Snort is an open source network packet monitoring and Intrusion Detection System. Snort looks for attack signatures, which are specific patterns of activity that has been defined to be of a suspicious or malicious intent. Snort analyzes network packets, and thus is classified as a Network Intrusion Detection System, or NIDS. These types of systems must be connected to the networks that they monitor and unless

the network topology is very simple, multiple Snort systems, called Snort sensors, must be setup and configured to monitor these networks. Snort relies on the ability to recognize attack signatures in order to identify an attack. These pattern recognition definitions are called rules. Attacks are not static, as they are continuously evolving as systems are protected to withstand existing attack methodologies. Thus, it is critical to perform analysis of prior activity to look for trends or changes in activity that are not typically classified as an attack which are often the precursor to an attack. Though it is possible to analyze information on multiple Snort sensors one at a time, it is difficult to summarize or perform analysis from a multi-Snort sensor perspective. For example, if an organization has multiple office locations in widely different geographical locations, it would be expected that separate Snort sensors are configured and operating. If an attacker targets the organization, it is possible that these different geographical locations are probed and attacked in series or simultaneously. Being able to recognize probing or attacking at a multi-geographic perspective can provide value in understanding individual sensor alerts. Snort evaluates data at the packet level and thus must process a large amount of data in real-time. Because of this, logging performance is important and to achieve this, the data storage implementation for Snort is optimized for fast writing. This results in a highly normalized database design where there are many one to one relationships. In fact, information representing the primary type of information in Snort, called an event (which is the packet information that matched one or more Snort rules), is represented in no less than six tables. One of the tables contains information that must always be provided for every event, and the other five tables contain information that is optional depending on the type of event that has occurred. This implementation allows for writing the smallest amount of information at a time, which allows for high performance when logging information. However, this design's drawback is when there is a need to read the information for reporting and other analysis. Displaying information for a single event may require joining six or more tables using outer joins, which impacts reporting performance.

Even with this highly normalized database design, the log data cannot be kept indefinitely, requiring that the data is removed from the sensor system by deleting older data. This results in the loss of data that could be used to develop better rules or provide evidence of an attack. Though it can be archived before deleting, the data is then offline and harder to analyze. Existing multi-Snort log reporting applications do exist. ACID, a popular web-database application has been available since. However, ACID is designed so that it can be configured as the primary store

for Snort log data and thus is subject to the same performance and historical data issues that the Snort sensors face – in the section titled “The Ongoing Use of the ACID Console,” Cox and Gerg [7] discusses deleting Snort log data on a periodic basis, though recommending backing up the data to some type of offline storage before deleting the data.

### 3 Snort Mart Design

The snort system uses various rule-based techniques based on comparison of past and current attacks. In order to implement efficient intrusion detection system a data mart integrated with snort is required. The real time data collection process is very intensive and produces large amount of data. In addition the whole system depends on data and database failure must be prevented. “Recognizing whether two sensors see the same or two different objects is a major challenge. Another challenge is effectively processing data streams that come from multiple sensors. Features that are not typical of the traditional database management system, such as almost real-time response, high input data rates, specific structure of incoming data, etc., validate the need for a sensor database.” [9] While Snort is focused on high performance logging, Snort Mart is focused on easy navigation and high performance reporting. To accomplish this goal, the database design for SnortMart emphasizes minimizing the number of entities and joins. One entity will contain the core information from the Snort sensors, while all of the other entities will contain classification information that will help guide queries to the core information. According to Bonifati et al, [5] data warehouses and data marts are becoming important asset of any modern enterprise, but despite the existence of several methods and tools addressing the problems related to the physical implementation of data marts, or to the manipulation of its data, little attention has been paid to the design of the data mart schema. The definition of the schema is one of the most critical steps in the overall data development process. In this project data mart is designed as a star schema using the 3-step design method proposed by Bonifati et al: top-down analysis, bottom-up analysis, and integration. The Snort database design defines the lowest level of detail as an event, which is the combination of a collection of packet data, called the data payload to an active Snort rule, called a signature. We use this level of grain for the SnortMart fact entity – a single row in the fact entity will be defined as a Snort event which consists of the Snort Sensor Event ID, Signature, Packet Data, IP, TCP, UDP, ICMP, and the date and time of the event. As mentioned previously, the Snort database design emphasized write speed and thus was implemented as a normalized relational database, using no less than six

entities to store the event, data, IP, TCP, UDP and ICMP information for each event (Snort Event, Network and Data Database Design). In addition, data types were stored in the most efficient data types rather than the more human readable formats. The six Snort entities were merged into a single fact entity in Snort Mart (Snort Mart Event, Network and Data Database Design) that utilizes more easily searched and human readable data formats. A negative consequence of this merging and data conversion resulted in that each fact row is quite large, which is not typical for dimensional data warehouses. The best fact attributes are typically numeric and additive and in Snort Mart we have the unique experience that none of the attributes fall into that category.

### 4 Snort Mart Implementation

The Snort sensors monitored two network devices, a NAT Router/Firewall and a web server. The NAT Router/Firewall was configured to allow Internet access to the web server, by mapping selected ports to the web server behind the NAT Router/Firewall on the internal network. This configuration was selected to allow a single attack to simultaneously attack the NAT Router/Firewall and the web server so we could generate Snort events that had identical timestamps to ensure that we could successfully merge data from multiple snort sensors with identical timestamps [6, 15]. The web server was an Intel-based PC running Microsoft Windows 2K Server, Microsoft IIS with several active websites and Windows Media Services. The attack system was an Intel-based PC running Fedora Core 4 (FC4) GNU/Linux, nmap (included with FC4) and Nikto 1.34. Nmap was selected as it was considered by Cox and Gerg [7], as one of the most widely used port scanners for network analysis. Nikto was selected as it was easy to use and freely available. Nmap and Nikto were used to supplement the real live probes and attacks to the NAT Router/Firewall and web server so that we could have enough Snort event data for testing. Each Snort sensor was an Intel-based PC running Microsoft Windows 2000 Professional, WinPCap 3.0, Snort 2.3.0 and MySQL 4.3.10. Each Snort sensor system had two Network Interface Cards, one connected to the network to monitor and the other connected to network consisting of the Snort Sensors and the Snort Mart system. Each Snort sensor was configured with identical rule sets (the set of rules included with Snort 2.3.0), to run in Intrusion Detection System (IDS) mode, and to log to the MySQL database engine installed on each Snort sensor. As indicated by Beale et al [4], logging to a relational database was selected as it is considered to be more efficient than logging to files. As described previously, the Snort Mart database model is

different than that of Snort. Both entity and attribute definitions are different, and thus must be processed before it can be added to SnortMart. This process of obtaining data from Snort, modifying it and loading it into SnortMart is called the Extraction, Translation and Load process. Though it is possible to attach to each Snort system, extract, translate and load the data directly into the Snort Mart database, this is not a desirable approach for ETL processing. This is because to perform the whole ETL process all at once per Snort system will require a longer time period in which the database connection between the Snort Mart and Snort system is open, which negatively affects the performance of the Snort system, as it has to maintain two database connections that will be accessing the same database tables. The implemented ETL process performs the extraction process, extracting each table to memory, disconnecting from the Snort database, and finally connecting to and saving the data into the Snort Mart ETL database. This method is used to minimize the load on the MySQL database engines used for Snort logs. Since both the extraction and translation/load processes shared some functionality, a shared script object for the Extraction and Translate/Load process was created – the object is instantiated in both the Extraction and Translate/Load scripts.

The system is implemented using Open Software whenever possible such as Snort, MySQL etc. According to Ashenfelter [1] MySQL is taking a serious role in the data warehouse. With its speedy execution, extensive set of built-in functions, and solid data import capabilities, it has all the relevant features of competing proprietary databases. Our system is currently in ‘prototype’ version. It collects and generates event, session, and full content data using Snort. Client-server architecture allows for running on Linux systems and client on Windows systems (typical administrator desktop) Report [15] provides instructions on installing the server components on a Red Hat server from scratch, and running the client on the same system or any Windows client supporting libraries. An early implementation [14] of this system as was using various databases systems for testing, but with new features in MySQL 5 we are currently experimenting with MySQL.

## 5 Findings

We found the Snort Mart database design allows for more simplistic use of the SQL l when constructing queries. This is due to the reduction of entities, use of human readable data types, and addition of entity attributes. The reduction of entities allows for the elimination of complex SQL joins. The greatest example of this is found when retrieving all of the fact information associated with an

event, where six Snort entities were merged into a single SnortMart entity. The use of human readable data types allows for more meaningful data to be used when constructing queries and reviewing the results returned by them. A good example of this is when accessing the IP address associated with an event. IP addresses are stored as integer values in Snort, rather than the IP4 (or IP6) format that most users are accustomed with (see Table 1, Snort vs. Snort Mart Data Types).

Snort Query	Snort Mart Query
Select * From event Left Outer Join iphdr On event.sid = iphdr.sid And event.cid = iphdr.cid Where ip_dst = 3436236630	Select * From events Where ipdestination = '192.168.0.12'

Table 1: Snort vs. SnortMart Data Types

In addition to easier query construction, the use of a database separate from the operational databases used by Snort sensors provides an environment that encourages greater use, as a Snort Mart user does not have to be concerned about affecting logging performance of Snort systems. We feel that these benefits will help encourage reporting and analysis of intrusion detection data.

## 6 Future Work

As computer attacks become more and more sophisticated, the need to provide effective intrusion detection methods increases. Network-based distributed attacks are especially difficult to detect and require coordination among different intrusion detection components. We propose a solution that responds to such requirements. Based on Open Software such as snort and MySQL Data Mart NIDS is a powerful concept which may change the way we protect our network. Our implemented NIDS model is in fact a prototype and needs to evolve into more mature and efficient model. Future work should emphasize a revisit of database design. One of the key reasons that the entities have so many attributes, in current implementation, was the concern of including important attributes and thus having all data available. This resulted in the inclusion of practically all of the event data. We believe that a good approach for achieving this would be an expansion of the solution: including and consolidated version of the operational Snort database that is it is used in conjunction with NIDS for reporting and analysis.

## 7 References

- [1] John Ashenfelter, Data Warehousing with MySQL, <http://www.mysqluc.com/cs/mysqluc2005/view>
- [2] S. Axelsson. "Intrusion Detection Systems: A Taxonomy and Survey." Technical Report No 99-15, Dept of Computer Engineering, Chalmers University of Technology, Sweden, March 2000
- [3] D. Paul Benjamin, Ranjita Shankar-Iyer, Archana Perumal, VMSoar: A Cognitive Agent for Network Security, <http://csis.pace.edu/robotlab/pubs/VMSoarS.pdf>
- [4] Beale, Jay, 2004. Snort 2.1 Intrusion Detection, Second Edition, Syngress.
- [5] Angela Bonifati, Fabiano Cattaneo, Stefano Ceri, Alfonso Fuggetta, Stefano Paraboschi, Designing Data Marts for Data Warehouses in: ACM Transactions on Software Engineering and Methodology (TOSEM) Volume 10 , Issue 4 (October 2001), 452 – 483.
- [6] Richard Bejtlich, "The Tao of Network Security Monitoring: Beyond Intrusion Detection", 2004 by Addison-Wesley.
- [7] Cox, Kerry and Gerg, Christopher, 2004. Snort and IDS Tools, O'Reilly Media, Inc.
- [8] Kimball, Ralph and Moss, Margy, 2002, The Data Warehouse Toolkit, Second Edition, John Wiley and Sons.
- [9] Jacob Nikom , Real-time Sensor Data Warehouse Architecture Using MySQL, <http://www.mysqluc.com/cs/mysqluc2005/view>
- [10] Snort Users Manual, Caswell, Brian and Hewlett, Jeremy, 2003, Snort.org.[http://www.snort.org/docs/snort\\_htmanuals/html\\_nual\\_232](http://www.snort.org/docs/snort_htmanuals/html_nual_232)
- [11] Snort Database Plugin Documentation. Danyliw, Roman, 2002, Snort.org. August 2002. <<http://www.snort.org/docs/snortdb/snortdb.html>>
- [12] Williams, Hugh E., and Lane, David, 2004, Web Database Applications with PHP and MySQL, O'Reilly Media
- [13] E. Thomsen OLAP Solutions: Building Multidimensional Information Systems, Wiley, New York, NY (2000).
- [14] Mizushima, M., Network Intrusion Detection System, Senior Project, 2005.
- [15] Wasniowski R. Network Intrusion Detection System, RAW-RR-09-02, Report 2004.
- [16] Wasniowski R., Multi-sensor agent-based intrusion detection system, Proceedings of the 2nd annual conference on Information security, Kennesaw, Georgia pp: 100 – 103, 2005.