Data Base Support for Intrusion Detection with Honeynets

RICHARD A. WASNIOWSKI
Computer Science Department
California State University Dominguez Hills
Carson, CA 90747, USA

Abstract: - As computer attacks are becoming more and more difficult to identify the need for better and more efficient intrusion detection systems increases. The main problem with current intrusion detection systems is high rate of false alarms. In this paper we discuss our experience in analyzing benefits of honeynets for intrusion detection. Using honeypots provides effective solution to increase the security; it is also useful tool for network forensic. Our purpose for this work is to examine how to integrate multiple intrusion detection sensors and honeynets in the order to minimize the number of incorrect-alarms. We present a framework for designing honeynets based project for network security analysis and an examples of the framework.

Key-Words: - Intrusion detection, honeypot, honeynets, false alarm, database, snort.

1 Introduction

The purpose of this paper is to discuss implementation of prototype multi sensor based intrusion detection system with honeynets. We are especially interested in analyzing traffic that has an abnormal or malicious character and should prompt a closer look. A specific feature of the model is that the systems use multiple sensors and data mining to process log files. This reduces the overhead in a distributed intrusion detection system.

2 Background

There are several intrusion detection systems, and one of the most popular in public domain is Snort[4]. Snort looks for attack signatures, which are specific patterns of activity that has been defined to be of a suspicious or malicious intent. 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. As indicated by Cox and Gerg [4], 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. The primary reason for building a data mart or data warehouse is to develop an intelligent, consolidated view of enterprise information. But each year, a large number of business intelligence and data warehousing initiatives fail because of erroneous or incomplete data. Often, users ignore the importance of developing a data management strategy as part of their extract, transform, and load or data warehouse architecture. 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. (see Fig.1and Fig2)

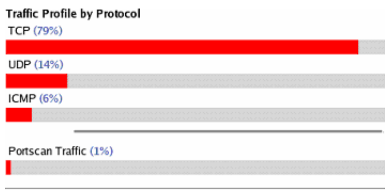

Fig.1 ACID monitor

Fig2 Snortalog example

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 [4] 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. A honeypot is a system whose value lies in being probed, attacked, or otherwise taken advantage of by attacker. Spitzner classifies honeypot solutions into two broad categories: production and research. For research purposes, we simply want to collect as much information on our attackers as possible. Production systems are generally used as an added layer of network security. Production honeypots are thought of as simpler and more intuitive than research honeypots. This affords system administrators the freedom to select from several commercially available (and sometimes free) honeypot solutions. Examples of such solutions that are currently available include BackOfficerFriendly, Specter, and honeyd. Research honeypots, on the other hand, are often homemade solutions that can track an attacker's actions. Network security professionals and educational institutions often employ research honeypots in the hopes of seeing a hacker in action. A honeypot that is to be used for research will often contain a fully operational operating system running certain services and vulnerabilities. Generally, this type of honeypot is much more difficult to configure and requires more time for upkeep. AIDE is a tool for trying to detect if someone has been on our machine and changed anything. If we know or suspect that someone has been on our machine, we can run aide to see what files have been modified, this will be a great help as we try to see what someone has done. AIDE works be creating a checksum of files in specific, user defined directories, saving these in a database, and checking them against the same files at a later date. The major drawback to AIDE is human, that is, it has to be run BEFORE an attack or it is worthless.

3 System Framework

Intrusion detection monitoring multiple systems and networks requires the existence of multiple intrusion detection systems. Each network being monitored requires its own intrusion detection system. Also, bandwidth limitations more than one intrusion detection system may be required on the same network.

In practical functioning intrusion detection systems produce false alarms called "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.

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. In order to be more responsive the intrusion detection system must be configured in a way that will probably report many of these "false positives". One possibility to minimize 'false positives" is to fuse data from multiple sensors. This requires both new data fusion methods and practical experiments. The main goal of our work is to develop such methods and test them in experimental setup. This report describes first step in designing and implementing such a system.

The snort system uses various rule-based techniques based on comparison of past and current attacks. In order to implement efficient intrusion detection system integration of multiple techniques and tools 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. The general framework for the system is shown on Fig.3, and Fig. 4.

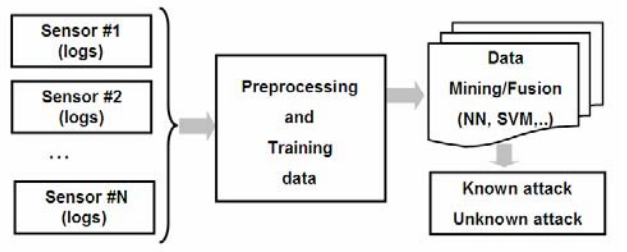
Sensor #1 (logs)
Sensor #2 (logs)
...
Sensor #N (logs)
Preprocessing and Training data
Data Mining/Fusion (NN, SVM,..)
Known attack Unknown attack

Fig.3

Fusion model similar to fwsnort [18] translates snort rules into an equivalent iptables ruleset. The Snort based multiple sensors system monitors two networks: One fully operational and one experimental. The NAT Router/Firewall is configured to allow Internet access to the web servers, by mapping selected ports to the web server behind the NAT Router/Firewall on the internal network. This configuration is 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. One web server was an Intel-based PC running Microsoft Windows 2003 Server, the second Centos based Linux system. The attack system is an Intel-based PC running Fedora Core 4 (FC4) GNU/Linux, laptop computer. Nmap was selected as it was considered by Cox and Gerg [4], as one of the most widely used port scanners for network analysis. Each Snort sensor is an Intel-based PC running CENTOS4.3 with Snort 2.3.0/2.6.0 and mySQL 4.3.10. Each Snort sensor is 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 [2], logging to a relational database was selected as it is considered to be more efficient than logging to files, and later logging to file was added as it is useful for analyzing data by some specialized packages such as Snortalog. The system is implemented using Open Software whenever possible such as Snort, mySQL etc. We are using Windows 2003 servers for our Web server and Honeypot. Our intrusion detection sensors are installed on Linux based systems.

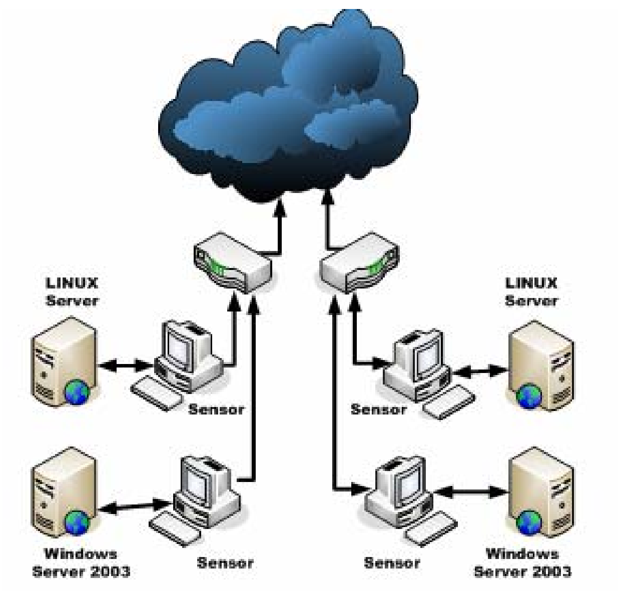

Fig. 4

4 Practical experiments

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. 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. On the whole, our information fusion based intrusion detection model is in fact a prototype and needs to evolve into more mature and efficient model. Future work emphasizes a revisit of database design to allow data fusion from multiple sensors.

6 Conclusion

In this work we have studied some data mining issues related to intrusion detection data, aiming at a complete data mining framework. In particular, we have justified the need for a data warehousing approach to handle intrusion detection data and we have focused on multidimensional access methods to efficiently index data. A second issue concerns clustering techniques on large alerts datasets.

References:

[1] J. Ashenfelter, Data Warehousing with MySQL, http://www.mysqluc.com/cs/mysqluc2005/vi

[2] Richard Bejtlich, “The Tao of Network Security Monitoring: Beyond Intrusion Detection”, 2004 by Addison-Wesley.

[3] A. Bonifati, F. Cattaneo, S. Ceri, A. Fuggetta, S. Paraboschi, Designing Data Marts for Data Warehouses in: ACM Transactions on Software Engineering and Methodology (TOSEM) Volume 10 , Issue 4 (October 2001), 452 – 483.

[4] Cox, Kerry and Gerg, Christopher, 2004. Snort and IDS Tools, O’Reilly Media, Inc.

[5] G. Giacinto, F. Roli and L. Didaci, Fusion of multiple classifiers for intrusion detection in computer networks, Pattern Recognition Letters 24 (2003), pp. 1795–1803.

[6] Y. Liao and V.R. Vemuri, Use of K-nearest neighbor classifier for intrusion detection, Computers and Security 21 (2002) (5), pp. 439–448.

[7] E. Lundin and E. Jonhsson, Anomaly-based intrusion detection: privacy concern and other problems, Computer Networks 34 (2000), pp. 623–640.

[8] MIT Lincoln Laboratory, http://www.ll.mit.edu/IST/ideval/.

[9] J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln

Laboratory, ACM Transactions on Information and System Security 3 (2000) (4), pp. 262–294
[10] J. Nikom , Real-time Sensor Data Warehouse Architecture Using MySQL, http://www.mysqluc.com/cs/mysqluc2005/vie
[11] Snort Users Manual, Caswell, Brian and Hewlett, Jeremy, 2003, Snort.org http://www.snort.org/docs/snort_htmanuals/htmanual_232
[12] Snort Database Plugin Documentation. Danyliw, Roman, 2002, Snort.org. August 2002. http://www.snort.org/docs/snortdb/snortdb.html
[13] E. Thomsen OLAP Solutions: Building Multidimensional Information Systems, Wiley, New York, NY (2000).
[14] Wasniowski R. Network Intrusion Detection System, RAW-RR-09-02, Report 2004.
[15] Wasniowski R., Multi-sensor agent-based intrusion detection system, Proceedings of the 2nd annual conference on Information security, Kennesaw, Georgia pp: 100 – 103, 2005.
[16] N. Ye, S. Masum Emran, Q. Chen and S. Vilber, Multivariate statistical analysis of audit trails for host-based intrusion detection, IEEE Transactions on Computers 51 (2002) (7).
[17] Teo, L., Sun, Y., Ahn, G. Defeating Internet Attacks Using Risk Awareness and Active Honeypots, Proceedings of the Second IEEE International Information Assurance Workshop (IWIA'04), 2004.
[18] Weiler, N. Honeypots for Distributed Denial of Service Attacks, Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02), 2002.
[19] Honeynet Project, http://www.honeynet.org/
[20] Computer Crime and Intellectual Property Section (CCIPS), 18 U.S.C. 2511, http://www.cybercrime.gov/usc2511.htm
[21] Yang, T.A., Yue, K., Liaw, M., Collins, G., Chen, P., etc. "Design of a distributed computer security lab". Journal of Computing Sciences in Colleges. Volume 20, Issue 1. pp. 332-346. The Consortium for Computing in Small Colleges, 2004.
[22] Spitzner, L. Honeypots Tracking Hackers, MA: Addison-Wesley, 2002.
[23] Provos, N. Honeyd: A Virtual Honeypot Daemon, Technical Report, Center for Information Technology Integration, University of Michigan.
[24] Provos, N. A Virtual Honeypot Framework, USENIX Security Symposium, August 2004.
[25] Provos, N. Developments of the Honeyd Virtual Honeypot, http://www.honeyd.org
[26] Baumann, R. Honeyd - A Low Involvement Honeypot in Action, originally published as part of the GCIA practical. http://security.rbaumann.net/download/honeyd.pdf
[27] Chandran, R., Pakala, S. Simulating Network with Honeyd, Technical paper, Paladiaon Networks, December 2003. http://www.paladion.net/papers/simulating_networks_with_honeyd.pdf
[28] Nmap, http://www.insecure.org/nmap
[29] Snort, http://www.snort.org
[30] Analysis Console for Intrusion Detection, http://acidlab.sourceforge.net/

http://www.tik.ee.ethz.ch/~weiler/papers/wetice02.pdf
[31]
[32] http://www.snort.org
[33] http://www.honeypots.net/ids/links
[34] http://online.securityfocus.com/infocus/1498