Analysis and Management of Intrusion Data Collection

Robert F. Erbacher
Department of Computer Science
Utah State University
Logan, UT 84322
Robert.Erbacher@usu.edu

Abstract

This paper expands upon our prior work [2] to examine the different data sources available for analysis in the identification of intrusions and misuses. Subsequently, we examine the different mechanisms by which data can be collected and the potential impacts this may have on the effectiveness of the analysis algorithms. Additionally, we examine the performance implications of such a data collection paradigm as is incorporated in our prior work. Examination and analysis of these performance impact results will aid determination of the most appropriate level of monitoring for a given environment or system. The goal is to identify information that must be collected everywhere versus that which should only be collected on critical systems and servers. By better arming systems and network administrators with the appropriate information they can make more adequate choices of their monitoring requirements and notify users of expected impacts and ramifications.

Keywords: Intrusion data, Security Management, Performance Metrics.

1. Introduction

Management of data collection and storage policies is quickly becoming a significant task for system and network administrators. The need to monitor system and network activity to aid in attack, intrusion, and misuse detection requires the acquisition and storage of enormous volumes of data. This is complicated by the need to collate all available information from within a distributed computing environment onto a single server for analysis. This collation is critical to identify global and distributed attacks, misuses, and threats and to bring all needed information within the auspices of the administrator.

When making decisions as to the volume and frequency of data to be collected administrators must weigh the tradeoffs associated with the volume of data collected and stored. Collect too little data and critical events could be missed. Collect too much data and the monitoring environment will in essence creates its own misuse of the available network accessible resources.

Identifying the appropriate level of tradeoff for a given environment is critical for the effective operation of said environment. For example, in a University setting where the typical activity on the network is in general diverse and chaotic, especially with students learning to create network-based programs for the first time, it would be ineffective attempting to monitor network activity. Additionally, in such an

environment it would be counter productive to reduce available network resources for monitoring purposes. The reduced performance of network-based applications would be deemed unacceptable.

Organizations which maintain sensitive data or rely on their network access for commercial activities would have an alternative set of requirements. For example, it would be unacceptable for a bank to have any type of unapproved access that might allow an intruder to gain access to information. Consequently, such organizations will have far more extensive monitoring requirements with network performance being of secondary concern to that of security.

The goal of this paper is to examine the performance measurement experiments we have executed, provide a context for these experiments within the scope of available data collection mechanisms, and examine the resulting impact on system performance such that more appropriate decisions can be made as to the extent of monitoring to be deployed. While extensive performance analysis has been done on TCP/IP networks in general [8], on the real-time monitoring of deployed routers and networks [13], and on the generation of real-time intrusion detection systems [7], no directed study has previously been performed on the implications of the collection of intrusion detection related data in a distributed network.

2. Data Collection Requirements

As discussed previously, the extent of data collection necessary for a given organization is directly related to the criticality of security within the specified organization. The data collection requirements will be dependent on what needs to be monitored, the extent of required monitoring, and the techniques used for the monitoring. There are six primary monitoring mechanisms, each can identify information and activity not available to the others and each is susceptible to problems and attacks the others aren't. The monitoring mechanisms include:

System statistics monitoring - System statistics monitoring incorporates system usage information over time, relating the volume of usage of various aspects of the system [11]. Such information can aid identification of saturated systems as well as possible misuse. Such statistics includes:

- **CPU load** - Identification of overall system usage. A high continuous load can be indicative of misuse, such as the running of a password cracker. In order to correctly relate the volume of activity the number of users present on the system is recorded as well.
- **Disk usage** - Indicates percentage and total disk usage, as with the `df` command. This can identify potential system failures due to lack of available resources, and potential misuse should available disk space change drastically in a short period of time.
- **E-mail message transmittals** - Indicates the number of e-mail messages sent and received through the specified system. Can be used to identify an open rely or virus infected systems.

System log monitoring - System log monitoring examines typical system log files. The information extracted from such files can include user connections and disconnections, e-mail message transmittals, initial TCP connection requests (with inetd -t), kernel and system messages, and application log events (such as portsentry alerts). These messages aid analysis of user behavior, e-mail behavior, failed connection history, as well as port scans and applications-based attacks. We have previously shown how these can be used in attack identification, especially in conjunction with system statistics [1, 2, 3].

Network flow monitoring - Network flow monitoring identifies connections and disconnections as with system log monitoring. Network flow monitoring examines the actual network traffic data [5, 10] and thus garners much more detail than is readily available from system log files. Network flow monitoring does not collect or examine the actual payload of any given packet but rather focuses on the header information. Additionally, network flow monitoring is less susceptible to attack as one of the primary goals of an attacker after they compromise a system is to compromise the log reporting facility. Network flow monitoring does have disadvantages including the need to locate the monitor (the location of which may be compromised), duplicate packet collection (particularly with broadcast packets), the inability to

associate users with specified traffic and thus an inability to monitor user behavior for misuse, and the added volume of data collected. Due to these limitations a hybrid approach using both network traffic monitoring and host-based monitoring is desirable.

Network data monitoring - Network data monitoring collects data similar to network flow monitoring but also examines the payload to what extent it is possible [5, 10]. The payload can aid identification of anomalous activity should known data streams be identified (such as buffer overflow attacks) or even unencrypted data streams that should be encrypted (e.g., passwords).

User process monitoring - The goal with user process monitoring is to examine the programs a user is running over time, creating a user signature [4]. This aids identification of misuse and intrusions based on anomalous behavior or behavior indicative of a misuse or intrusion. For example, intruders will use the 'who' command frequently to ensure they are alone on the system to avoid identification [9].

Library function monitoring - Library function monitoring extends user process monitoring to examine the functions that are actually being called by the programs the user is running and the sequence of such calls [6]. The idea behind this facility is that with user process monitoring the user can avoid detection by running modified or renamed copies of well known programs. For example, by renaming the `who` command they will avoid the characteristic behavior of its use. Library function monitoring essentially identifies the behavior of the given process to associate its behavior with that of known programs.

For the purpose of this paper we will focus on extensive system log monitoring and system statistics monitoring. Since for the purpose of this paper we are primarily concerned with measuring the performance implications of such monitoring, the susceptibility of system log monitoring to attack is not of relevance and it will provide indications of the impact of network flow monitoring as they are very similar in nature for the bulk of the data. Additionally, for the purpose of this research we are considering host-based monitoring facilities rather than router or switch -based facilities, such as mirrored switches.

3. Test Environment Configuration

The test environment consisted of a single server running Linux (kernel 2.4). The server is a dual processor Intel system running at 1 GHz, 1GB main memory, 60GB disk space. All data was stored into a postgres SQL 7.0 server [12]. While the environment is capable of collecting data from many distributed computing facilities simultaneously, for the purpose of these performance experiments we limited data collection to a single remote server. The remote server is a quad processor Ultra-80 sparc server running SunOS 5.9 with 32GB of main memory. The network connectivity between the two systems consists of a 100Mb network with 1 hop between the two systems across the university's primary backbone.

The data collection was performed by a standard Bourne shell script as well as a conversion of this same script into 'C'. Neither version of the data collection program was extensively optimized for either performance or memory usage. All of the data was plotted using xmgrace. The data collection is broken into seven sections or components exemplifying the type of data being collected, the mechanisms employed to collect it, and the source of the information. These include:

- **Uptime** – Uptime collects information on the system load. This incorporates the typical parameters indicative of the load average over the last 1, 5, and 15 minutes.
- **Mail** – Representative of the mail transmittal messages logged by the system.
- **DF** – Represents the path and usage parameters for each disk accessible from the system.
- **System** – Identifies system-based messages, including `portsentry` messages, kernel logs, etc.
- **Last** – Identifies user's connection and disconnection times.
- **Who** – Identifies the number of users logged into the system at the indicated time.
- **None** – Incorporates no actual collection mechanisms but rather is indicative of the overhead involved with merely running the data collection scripts.

4. Experiments

In this paper we are focusing on seven experiments to measure the performance impact of various activities in the data collection process. Chart 1 exemplifies the typical decision making process. In a typical data collection scenario we must be concerned with the sensitive nature of the data being collected. We already mentioned that the data is being stored in a postgres database. However, the SQL commands are currently transmitted in clear text. Given the potential for private information or information useful to an attacker to appear in such transmittals it is likely necessary to transmit the information in an encrypted fashion. Chart 1 shows the incorporation of an encrypted tunnel between the two machines ports through the incorporation of stunnel. Two mechanisms are incorporate, the first is a command-based mode in which the stunnel connection is opened and closed for each SQL command. Leaving a connection open persistently may have its own disadvantages and thus this mechanism avoids this. However, this mode has significant performance ramifications. The actual data points are plotted as red squares along with the linear regression as a single red line. The mode with no compression is plotted as orange triangles and the session-based mode is plotted as green circles. The session-based mode opens a single stunnel connection at the beginning of data collection and only closes it at the end of data collection. It is interesting to note that the session-based encryption mode actually encompasses less performance impact than with no encryption at all. This is counter intuitive as it would be expected that the incorporation of the encryption in and of itself would have significant ramifications. Essentially, the ability to avoid the cycles associated with the SQL commands opening a remote connection with the SQL server and instead making this connection essentially a local connection to the stunnel port more than makes up for the required encryption.

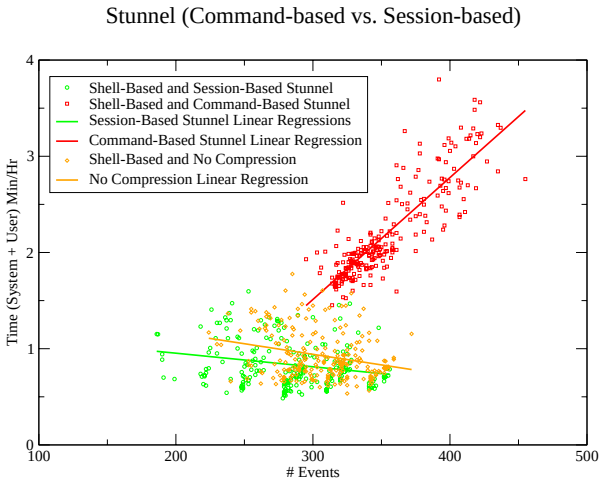

Chart 1: This chart compares the performance implications of using command-based versus session-based stunnel connections with no compression enabled.

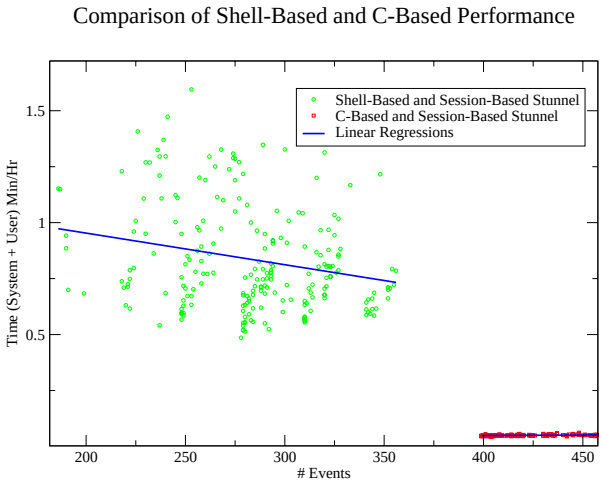

Chart 2: This chart compares the performance implications of using a shell-script versus a C-based program for data collection.

The second chart is simply a comparison of the performance ramifications of using a C-based versus a shell-based implementation. For many data collection needs a shell-based implementation may be more intuitive and easier to provide. In this chart neither implementation has been extensively optimized. Even with far more events, the C-based implementation has far less of a performance impact than the shell-based implementation. The varying clusters of events/performance impact in both chart 1 and chart 2 are indicative of greater numbers of different types of events occurring. We will examine the impact of each type of event in charts 6 and 7.

Chart 3 examines the impact of using SSH for the encryption protocol rather than stunnel. Given the impact of chart 1 we are limiting our analysis to session-based encryption. As with stunnel, SSH provides the ability to create an encrypted tunnel connection between two machine's ports. An encryption tunnel

essentially creates an encryption channel between a server and a client without the need to modify either. It encrypts all communication between the specified ports. While this is the primary and sole capability of stunnel, SSH provides additional extensive capabilities and tunneling is actually not even its primary function. The ramifications of these differences in goals are clear by the exhibited impact.

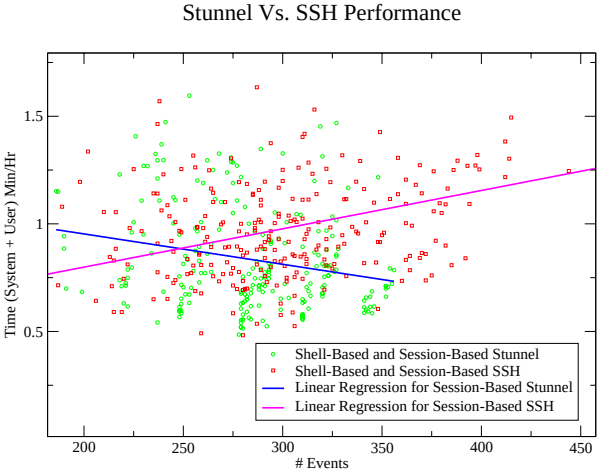

Chart 3: This chart compares the effectiveness of stunnel versus SSH for providing the encrypted connection.

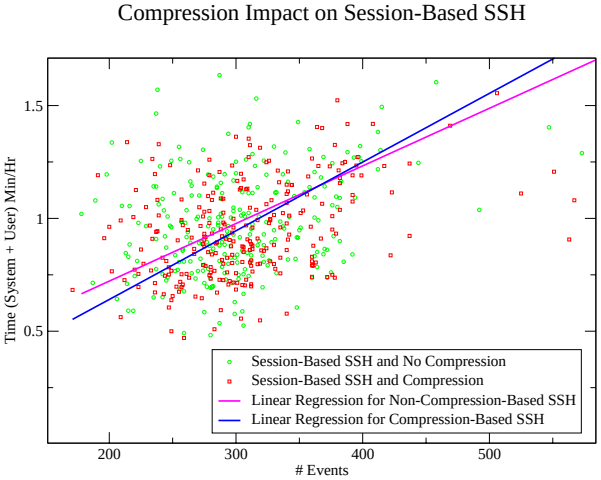

Chart 4: This chart shows the impact of enabling compression within the encrypted data stream.

Chart 5 shows the impact of different data collection frequencies. Typically we initiate a data collection every 300 seconds (5 minutes). While many of the data elements will be identical between the sets, such as user connection and disconnection information, other elements will differ. This includes system statistics, such as uptime, which will have far more information with higher collection frequencies. There is also overhead associated with each run of the data collection facility. Consequently, the impact of higher frequencies of collection doesn't quite have a linear impact.

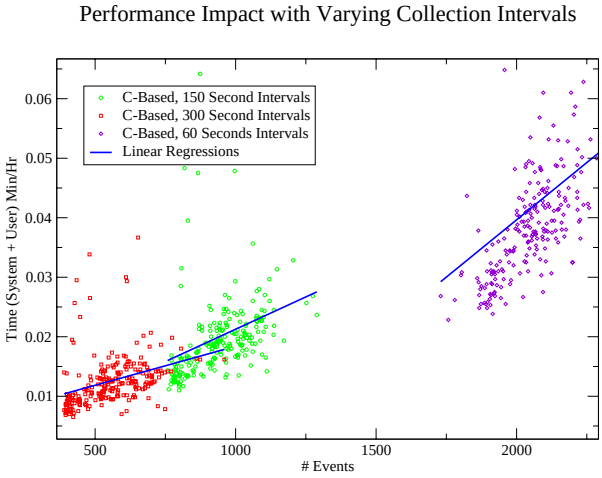

Chart 5: This chart shows the impact of various data collection rates ranging from initiating collection every 60 seconds, 150 seconds, to every 300 seconds.

Charts 6 and 7 show the impact of the individual collection mechanisms integrated into the data collection environment. Of note is the high impact of the disk usage reporting facility. Even though it executes the same number of times every time (the number of file systems rarely changes), the performance impact varies slightly. The file system reporting subsystem also incorporates the greatest impact of all the subsystems. The single advantage of this subsystem is that since the number of file systems does not

change the impact of this subsystem will not increase significantly even on much larger systems. The who and uptime facilities require identical amounts of time to perform. Even with very few events the mail subsystem requires enormous overhead to process the transmittal logs. While none of the individual event mechanisms, aside from `df`, individually incorporate substantial performance impacts the linear regressions imply that on large systems the impact of the system as a whole is going to be substantial, especially on a mail server in which large numbers of mail transmittals must be resolved.

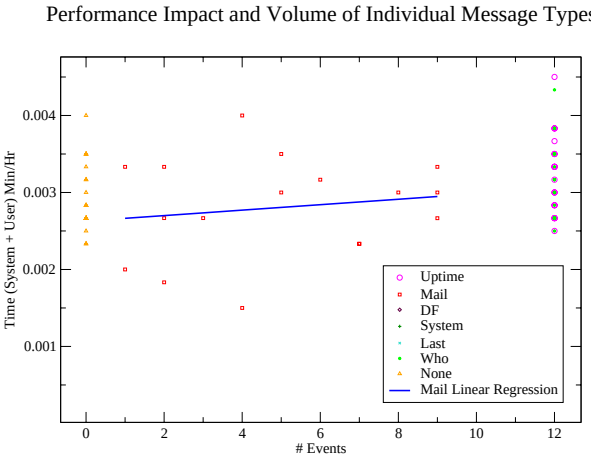

Chart 6: This chart examines the volume of each type of activity and the associated performance implications of each of those types of activities. More specifically, this chart shows the lower range of activity that appears occluded in **Chart 7**.

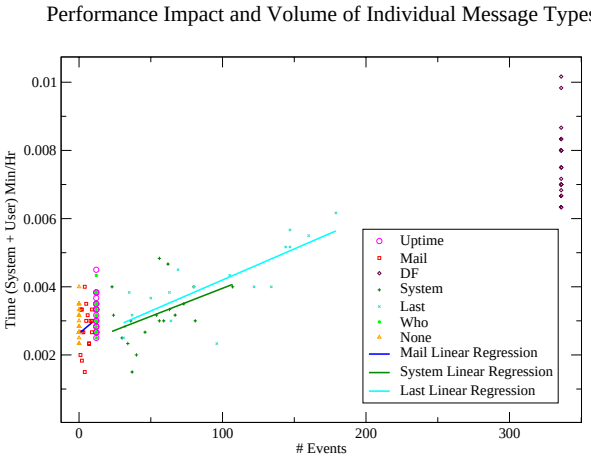

Chart 7: This chart examines the volume of each type of activity and the associated performance implications. The entire range is exemplified in this chart.

5. Other techniques

It should be noted that given the volume of data available, especially with respect to network traffic data, it is realistic to assume that not every piece of data crossing the network is going to be captured and stored. It is quite common for administrators to sample network traffic data sporadically to identify unusual activity, e.g., the presence of an unapproved IRC server. Such sporadic sampling is likely to identify continuous activity but may not detect an actual intrusion or compromised machine. The goal is to capture select subsets of the network as a whole in great enough frequency such that differentials between the system log data and the network traffic data can be used to identify a compromise. Two mechanisms by which this can be accomplished are frequency sub-sampling, and statistical sub-sampling.

5.1. Frequency sub-sampling

The benefits of frequency sub-sampling are the simplicity and consistency of the algorithm. Essentially, every n^{th} packet will be collected from the network. This is effective at reducing the shear volume of data needing to be collected, but the unscientific method by which the data is collected will allow anomalous activity to go unnoticed. In fact, hackers aware of such an implementation can easily initiate countermeasures, ensuring that their packets (or a majority thereof) are not collected. The predictability of this mechanism reduces its effectiveness. Its ease of implementation is its principal advantage.

5.2. Statistical sub-sampling

In order to improve upon the frequency sub-sampling algorithm we can implement a more randomized collection approach in order to limit the ability of hackers to avoid detection through knowledge of the technique. By collection of a randomized but statistically significant sub-sample we can both reduce the volume of data collected and implement a technique that will collect sufficient details of an anomalous event to sufficiently identify it. The difficulty is in its implementation and more specifically the identification of a statistically significant sub-sample. Collect too little data and an attacker will go unnoticed. Collect too much data and the volume of data collected won't be sufficiently reduced.

6. Conclusions and Future Work

We have examined available data collection mechanisms and their impact on the effectiveness of intrusion detection and misuse algorithms. Additionally, we have examined the performance impact of some of these data collection facilities and the impact of different management choices that must be made during the deployment of such data collection facilities. The impact of many of these decisions is non-trivial and unexpected. Through the dissemination of these experimental results we hope to allow system and network administrators to be able to make better decisions with respect to their data collection schemes as applied to intrusion and misuse detection.

While we feel these results will go a long way towards accomplishing our specified goal we are aware of several weaknesses in the current experiment set. For instance, we do not incorporate any analysis of network bandwidth. Often network bandwidth is more important for system administrators than the impact on each system's CPU utilization. This is in part due to the fact that data collection from all available systems will accumulate their network usage, especially on network segments connected directly to the server where the information is being stored. Additionally, we must provide more extensive analysis and comparison of different system types. For example, the impact of the data collection on servers as opposed to workstations must be examined in greater detail. Also, while our data collection facilities currently run on UNIX-based systems we must examine the implications of such data collection facilities when carried over to Microsoft Windows-based systems.

7. References

1. Robert F. Erbacher, "Intrusion Behavior Detection Through Visualization," *Proceedings of the IEEE Systems, Man & Cybernetics Conference*, Crystal City, Virginia, October, 2003, pp 2507-2513.
2. Robert F. Erbacher and Bill Augustine, "Intrusion Detection Data: Collection and Analysis," *Proceedings of the 2002 International Conference on Security and Management (SAM '02),* Las Vegas, NV, June 2002, pp. 3-9.
3. Robert F. Erbacher, Kenneth L. Walker, and Deborah A. Frincke, "Intrusion and Misuse Detection in Large-Scale Systems," *Computer Graphics and Applications*, Vol. 22, No. 1, January/February 2002, pp. 38-48.
4. S. Freeman, A. Bivens, J. Branch and B. Szymanski, "Host-Based Intrusion Detection Using User Signatures," *Proc. Research Conference*, RPI, Troy, NY, October, 2002.
5. V. Jacobson, C. Leres, and S. McCanne, "tcpdump," Lawrence Berkeley National Labs, ftp://ftp.ee.lbl.gov/, 1989.
6. Kuperman. B. A. and Eugene H. Spafiord, "Generation of application level audit data via library interposition", CERIAS TR 99-11, COAST Laboratory, Purdue University, West Lafaytte, IN, October 1998.
7. K. Ilgun, "A real-time intrusion detection system for UNIX," *IEEE Symp. on Security and Privacy*, 1993.
8. T. V. Lakshman and Upamanyu Madhow, "The Performance of {TCP/IP} for Networks with high Bandwidth-delay Products and Random Loss," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 3, 1997, pp. 336-350.
9. Linda McCarthy, *Intranet Security: Stories from the trenches*, SUN Microsystems Press, 1998.
10. M. Roesch, "Snort - lightweight intrusion detection for networks," *13th Administration Conference, LISA'99*, Seattle, WA, Nov, 1999, USENIX.
11. Eugene Spafford and Diego Zamboni, "Data Collection mechanisms for intrusion detection systems," CERIAS TR 2000-08, COAST Laboratory, Purdue University, West Lafaytte, IN, June 2000.
12. M. Stonebraker and G. Kemnitz, "The postgres next-generation database management system," *Communications of the ACM*, Vol. 34, No. 10, 1991, pp. 78-92.
13. Http://people.ee.ethz.ch/~oetiker/webtools/mrtg/