PERFORMANCE EVALUATION OF SNORT IN AN INEXPENSIVE HIGH-AVAILABILITY SYSTEM

Wan Kyung Kim*, Woo Young Soh, Jason S. Seril*

Department of Computer Engineering, Graduate School, Hannam University, Korea,
*(SSPSC-SSIT Campus, Surigao, Philippine)**
wankk12@neuro.hannam.ac.kr

Abstract—**Most studies in the past in testing and benchmarking on Intrusion Detection System (IDS) were conducted as comparisons, rather than evaluation, on different IDSs. This paper presents the evaluation of the performance of one of the open source IDS, snort, in an inexpensive high availability system configuration. Redundancy and fault tolerance technology are used in deploying such IDS, because of the possible attacks that can make snort exhaust resources, degrade in performance and even crash. Several test data are used in such environment and yielded different results. CPU speed, Disk usage, memory utilization and other resources of the IDS host are also monitored. Test results with the proposed system configuration environment show much better system availability and reliability, especially on security systems.**

I. INTRODUCTION

This paper designs and implements a high availability system configuration for network intrusion detection system and benchmarks **snort** version 2.0 as the subject, its performance in such environment and how it can be optimized. This benchmarking is used to gather empirical results in the form of performance data and evidence, of both the new system's effectiveness and usefulness. The objectives in this paper are to design a low-cost high availability system configuration for network intrusion detection system(NIDS) using **snort**, to evaluate the system performance to better understand how much system resources is used by **snort** in such a system environment and how **snort** can be affected by "stressed" conditions in the computing environment.

One of the objectives of this study is a design of a fault-tolerant NIDS in a two-node environment. This framework comprises functionally redundant systems that provide reliable service despite NIDS failure. Fig. 1 shows the top-level design framework of **snort** in a fault tolerant environment.

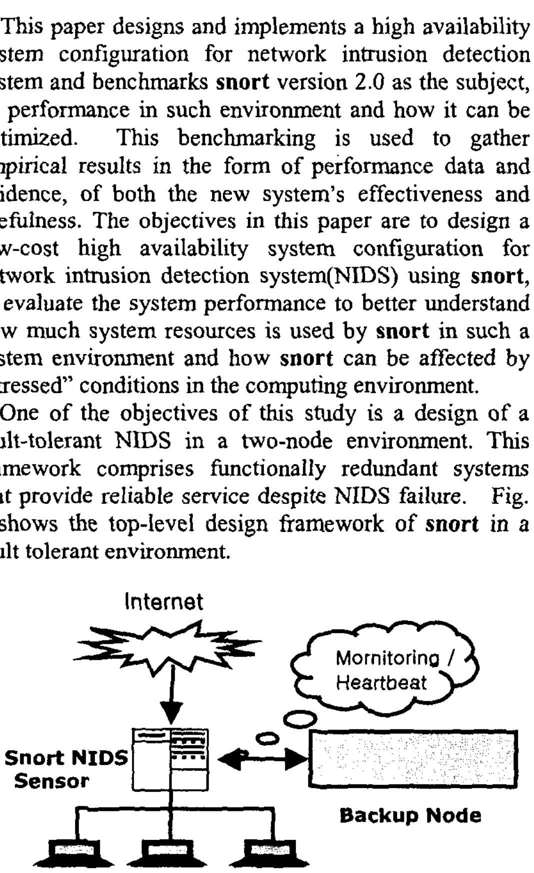


Fig. 1 Top-level Design of Sort NIDS in a high availability machine

As stated in **snort**[1] is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. Aside from it, **snort** can perform also protocol analysis and content searching/matching in order to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, PS finger printing attempts, and much more. **snort** has real-time alerting capability: alerts can be sent to syslog, Server Message Block, Database, or a separate '*alert*' file.

snort detection engine relies on rules to detect intrusions, and it cannot maintain states. Several preprocessors are introduced to keep states and deal with portscan, IP fragments and TCP fragments. These preprocessors are run before the detection engine is called and after the packets have been decoded.

Fig. 2 depicts the **snort** architecture, which has three principal components.

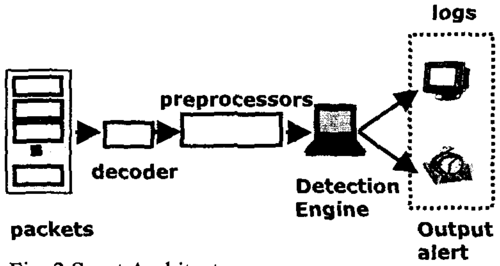


Fig. 2 Snort Architecture

In 1999, the U.S. DARPA initiated the latest evaluation on IDS. It is considered as the most comprehensive scientific study for comparing the performance of different IDS. The network data generated from its private controlled network are used to evaluate IDS. The data is analyzed off-line to determine which sessions are normal and which constitute intrusions. Although this is beyond the scope of the study, more of these attacks can be found on these [2][3] conducted.

The most cost-effective approach to increasing one's systems reliability is to implement a fail-over configuration. Fail-over setup involves pooling together multiple computers, each of which is candidate server for a file systems, databases or applications. Each of these systems monitors the health of other systems in the cluster. In the event of failure in one of the cluster members, the others take over the services of the failed node. The takeover is typically performed in such a way as to make it transparent to the client systems that are accessing the data.

To have a failsafe intrusion detection system, Linux High Availability (HA) machine were set-up, using several open source packages available. Virtually every UNIX supplier has their own HA software solution to provide customers with fail-over server systems at moderate prices or even free of cost. To come up with such fail-over environment, several HA solutions were used. Such environment is discussed below.

II. TESTING METHODOLOGY

The experiment described in this paper uses a simulated network and **snort** 2.0 network intrusion detection system. The IDS is evaluated using the captured sample **tcpdump** traffic from MIT's Lincoln Lab IDS evaluation performed in 1998 [4]. This sample data contains only simple attacks, which are included to illustrate how intrusion detection system will be evaluated. Attacks include instances where a remote user illegally obtains local user-level privileges or local root-level privileges on a target machine and instances where a remote user surveys a potential target for weaknesses or searches for potential targets. Attacks in the sample data include the following:

Table 1 Different Attacks in DARPA Sample Test Data

Name	**Description**
Guess	Remote user guesses many passwords to log into a target machine.
ping-sweep	Low level ICMP ping sweep to identify target machines.
port-scan	Determine which services on a target machine are active.
phf	Run Unix command line on a web server.
Rlogin	Rlogin to target machine without a password.
Rsh	Execute a command on the target machine without a password
Rcp	Remotely copy a file to/from target machine without a password.

With the sample test data mentioned above, there are 272 non-intrusive sessions and 39 intrusive sessions.

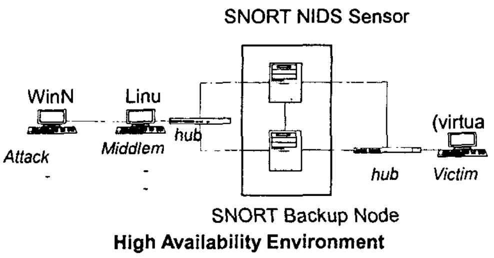


Fig. 3 Testbed network

Fig. 3 depicts the test bed environment. And to simulate back the capture data from DARPA, **tcpreplay** tool was used. A tool for replaying network traffic from files saved with **tcpdump. Tcpreplay** is being played on the "*middleman*" computer, and the saved packets were being sent to **snort** sensor. Another test involve is injecting packet onto a test network on which the subject **snort** was running. On this test, it engages the TCP protocol. In some cases, the tests have some interacting between the injected packets and the third host, representing a hypothetical "*target*" of attack. In each test, this target host was the exploit addressee of the entire packet injected.

In testing or evaluating snort, sets of performance objectives were identified first(which are similar to the design goals cited in [5]):

- ☐ Broad Detection Range
- ☐ Economy in Resource Usage
- ☐ Resilience to Stress

Fail-over Test : Experiments have been performed to ensure the high availability of **snort**. Several of these tests were performed like taking down the primary host, kill its power, and the *eth0* heartbeat cable.

Intrusion Identification Test : Intrusion Identification Tests that measure the ability of **snort** IDS to distinguish known intrusions from normal behavior. Some baseline tests were carried out first on **snort**. This test is discussed on the following pages.

Basic Detection Test : Before conducting complicated or subtle tests against the subject, the researcher conducted a series of "*baseline*" tests. The purpose of these tests was to ensure that the subject IDS, was configured properly and was functioning at the time this tests were conducted and that **snort** did in fact detect attacks from the test data.

The researcher employs the **sidestep** package[6] to inject attack packets in the testing environment. The **sidestep** package is based on MS Windows; it sends attack at the target that evades an IDS. This package provides novel attacks such as *phf, dns, rpc, ftp*, and several other attacks. It has three different modes of attacks to the victim, the *normal, evade* and *the false alarm* attacks.

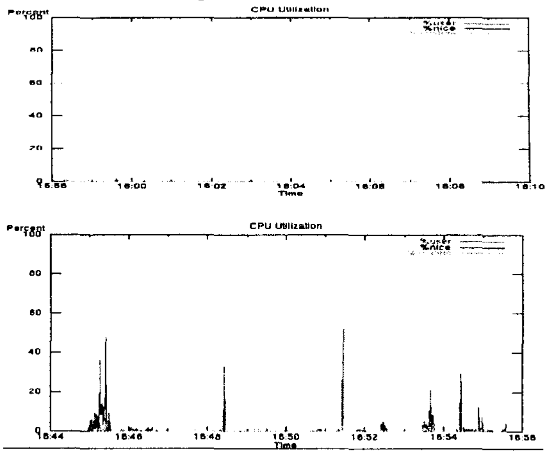
Economy in Resource Usage : This test is conducted to measure how much system's resources are used by **snort** IDS. This is to decide if it is practical to run **snort** in a particular computing environment, such as in fail-over environment. To measure such activities of the host including the nodes, **sar** (system activity reporter) tool has been used. This command writes to standard output the contents of selected cumulative activity counters in the Linux operating system. The accounting system, based on the values in the *count* and *interval* parameters, writes information the specified number of times spaced at the specified intervals in seconds. Resilience to Stress : One of the performance objectives of this evaluation is to know whether **snort** could still function correctly under stressful conditions in the system, such as a very high level of computing activity. Like for instance an attack that **snort** would ordinarily detect might go undetected under such conditions. There are several forms of stress test conducted on **snort**. These tests were developed by [5]. One of the stress test being used is the load test. This investigates the effect of the load on the **snort** host CPU. In this testing experiment, the researcher measured the load of the CPU by using **sar** command, which again reports the average number of jobs in the CPU run queue.

This test is conducted in the same way as the other tests conducted, except that a high load should be established on **snort** host during test. A high load can be created by running additional programs on the **snort** host, so that **snort** program must share CPU time with the other programs.

III. EXPERIMENTAL RESULTS

Fail-over Test : There were two tests conducted to check such setup. The results are discussed on the following sections.

Takeover Test : During testing, few scenarios have been deployed to check whether high-availability is attained. One situation was to run the **snort,** get its process, kill it and verify whether the back-up node take over as the sensor. Another condition tested was when the **snort** host was shutdown. The results were all as expected. A crossover cable (*eth0*) was used for the monitoring of ones "heart beats". In the case of the inter-node/heartbeat network failing, the nodes simply carried on normal operation and do some alerts. Sensor Node's Performance : One of the benchmarks measured the overall **snort** host performance including CPU average load, memory statistics and I/O transfer rate during the subject IDS **snort** is running with and without the backup node. Fig. 4(a&b) visually shows the average performance of the **snort** host without and with the backup node. And it depicts that the overhead of the host is not that high.



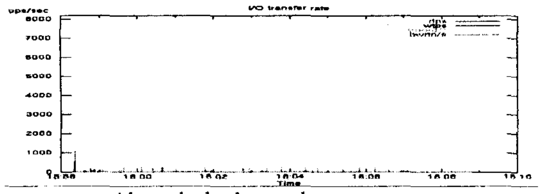
a. without the backup node
b. with the backup node

Fig. 4(a & b) Average Performance of the host with SNORT running

The results show that the CPU utilization of the snort sensor node varies when backup node constantly monitors the "*heat beat*" of the said sensor. Basic Detection Test : For this test, the output of snort that are being logged and saved to the database using the tool ACID from the "*baseline*" test conducted, were being analyzed. As mentioned earlier that with this test, it is to ensure that snort is properly configured with its plug-ins, particularly its preprocessors and rules, and to check if snort reacted to the test by either reporting or not reporting the sidestep attacks. By considering the snort's output and the specific attack packets used for the test, the researcher was able to deduce significant characteristics of the said IDS. Resource Usage Test : At this point, the researcher tested the snort host's resource usage. During the actual test using the 1998 DARPA's sample data, the total disk space used by snort's output through ACID was measured by using the UNIX sar (system activity reporter) command.

Fig. 5(a&b) shows the graph of the I/O and transfer rate statistics without the backup node. The measurement have been made during the running of sample test data from 1998 DARPA IDS evaluation. With this report, total number of transfers per second that were issued to the physical disk was collected. Aside from this data, the following reports values are displayed. The total number of read requests per second issued to the physical disk, as well as the write requests per second issued to the physical disk are displayed.

Comparing the result in the first environment where the backup node was disabled, Fig. 5(a&b) shows that the total amount of data read from the drive in blocks per seconds change as the backup node continuously checks the snort sensor node.


a. without the backup node

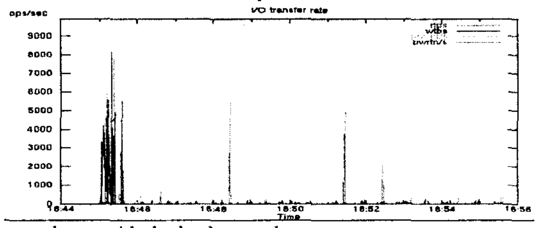
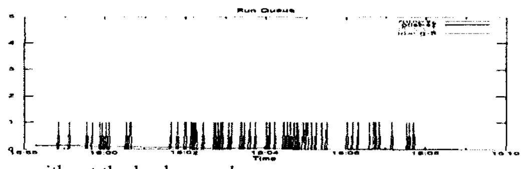

b. with the backup node

Fig. 5(a&b) Graph of I/O and Transfer Rate Statistics without the backup node

Stress Test : As discussed in Chapter 4, CPU run queue of **snort** host is measured in such stressful conditions. The researcher hypothesized that such stress in the form of high load on the **snort** host might affect its ability to monitor network connections. Fig. 6(a&b) below visually shows the **snort** host CPU load, the average number of jobs in run queue without and with the backup node.


a. without the backup node

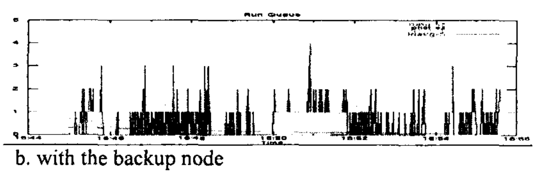

b. with the backup node

Fig. 6(a&b) SNORT Host jobs in run queue vs. Time

This also reports the run queue length, which is the number of processes waiting for run time. The number of processes in the process list, the system load average for the last minute and the system load average for the past five (5) minutes were also displayed in the above graph. Aside from this report, memory activities and swap space utilization statistics were also measured. Values of such statistics were taken from **sar** tool. Fig. 7 shows the graph of the memory activity where backup node is enabled. First the amount of free memory available in kilobytes during the test being ran. The amount of used memory in kilobytes, percentage of used memory, the amount of free swap space in kilobytes and several other values are being displayed.

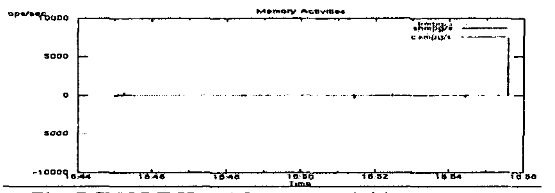
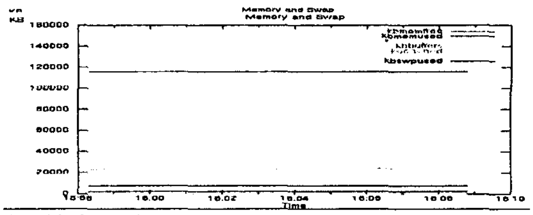


Fig. 7 SNORT Host Memory Activities

The figures on the next page graphically show the memory and swap statistics. Fig. 8(a) portrays the utilization information where the secondary node is immobilized, and in Figure 8b shows a different data where the said node (backup) where enabled.

a. without the backup node



b. with the backup node

Fig. 8(a & b) Memory and Swap Space Utilization Statistics

IV. CONCLUSIONS

Evaluating the performance of an NIDS such as **snort** in a high availability system configuration is a challenging task. Several factors have to be considered, and these factors will change from situation to situation. As the growth in the use and development of NIDS continues, such testing techniques are growing in importance. This paper presented a network environment of a fail-over network intrusion detection system in a high availability machine. The results from the study show the viability of such approach for redundant IDS. This paper presented an inexpensive high-availability solution design for the mission-critical needs without requiring the use of expensive additional hardware or software. On the whole, this two-node system configuration environment provides with much better system availability and reliability, especially on security systems. It is a vast improvement over the single node, as it is affordable to do server maintenance.

Future directions of this research would include the design and implementation of a multi resource high-availability environment such as redundant database for the IDS alert logs, file system takeover and others are also planned to be undertaken as future research.

REFERENCES

[1] Snort's Documentation, URL: http://www.snort.org

[2] K. Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", M.Eng. Paper, MIT Department of Electrical Engineering and Computer Science, June 1999.

[3] K. J. Das, "Attack Development for Intrusion Detection Evaluation", M.Eng. Paper, MIT Department of Electrical Engineering and Computer Science, June 2000.

[4] Richard Lippman, *et. al.*, "*The 1999 DARPA Off-Line Intrusion Detection Evaluation*", submitted to Proceedings of 3rd International Workshop on Recent Advances in Intrusion Detection (RAID 2000).

[5] N. Puketza, et. al., "*A Methodology for Testing Intrusion Detection System*", Proc. 17th National Computer Security Conference, October 1994.

[6] http://www.robertgraham.com/tmp/sidestep.html