Managing Vulnerabilities in a Networked System

Muhammad Ali Brohi*
[1a] Faculty of Science Department of Computer Science, Northern Borders University, Arar, Saudi Arabia
[b] Information Communication Processing Center, Mehran University of Engineering & Technology, Sindh, Pakistan

Jamshed Mustafa Khan

[2] Department of Electrical Engineering Northern Border University Arar, Saudi Arabi

Mahera Erum Baloch
[3] Institute of Computer Engineering, University Duisburg-Essen, Campus Duisburg, Germany.

Abstract

Intrusion detection systems are not easily constructed or maintained due to the almost daily evolution of network traffic and known exploits. The research in this paper evaluates it through analysis of the documentation published for the University Network as well as experimentation using different rule customizations.

Snort is selected because of its price and easy customization through the manipulation of its rules files. This shows that this benchmarking system can be easily manipulated. Developers looking to enhance performance can alter their rules files to better detect attacks. This system can be manipulated to produce better results, and thus becomes less a test of developers testing their true systems and more a test of how well developers can interpret the testing data.

The research in this project shows that benchmarking the intrusion detections systems cannot be carried out effectively at this time. Unless we develop a more advanced artificial intelligence and data mining technique, it will be very hard to evaluate the intrusion detection systems. The amount of customization that goes into effectively using one, as well as the ever-changing number of viable network exploits makes it impossible at this time.

***Keywords*:** *Vulnerabilities*, Snort, security, Threats, IDS/IPS, History.

1. Introduction

Network security is a thriving industry in this country as more and more of the corporate workspace is converted to digital media every day. Because companies and home users keep sensitive information on their computers, there is a great need to protect that information from those who would exploit it. One way to help keep attackers at bay is by using an intrusion detection system (IDS), which are designed to locate and notify systems administrators to the presence of malicious traffic. The current systems are not effective right now because detecting intrusions and other forms of malicious traffic in a large, modern corporate network is difficult [1]. Something must be done in order to improve performance and make these systems ready for reliable operation in a dynamic environment.

We can classify IDS's as host-based and network-based. Host-based intrusion detection system monitors the computer. The software is running on and often integrates closely with the operating system. Network IDS/monitors network traffic between the hosts. Unlike a host-based system, which detects malicious behavior outright, these systems deduce behavior based on the content and format of data packets on the network [2]. This project looks exclusively at network-based intrusion detection systems, as opposed to the host-based intrusion detection systems.

A reliable and efficient intrusion detection system (IDS) is a necessary component in any network. It can alert administrators of possible attackers and give a good view of the network's status [3]. This section looks at current systems, proposals for new types of IDSs, and higher level ideas that could be carried over into IDS development. It is the main goal of this project to look at how this IDS/IPS performs in a real-world environment.

The IDS looked at most closely in this project, Snort, is a rule-based network intrusion detection system (NIDS) [4]. Martin Roesch, in his paper entitled "Snort – Lightweight Intrusion Detection for Networks," says "Snort fills an important 'ecological niche' in the realm of network security: a cross-platform, lightweight network intrusion detection tool that can be deployed to monitor small TCP/IP networks and detect a wide variety of suspicious network traffic as well as outright attacks" [5]. The SANS Institute also reported Snort as becoming the standard among intrusion detection experts due to the fact that it is open-source, frequently updated, and free of charge [6]. Snort generates a number of false positives, which can amount to thousands per day on a network attached to the Internet running a default installation of Snort [7].

Main purpose to take up this research was to improve the overall quality of intrusion detection systems by

analyzing the current ways to test these systems. Many "false positives," which is standard traffic being diagnosed as malicious data, and sometimes a few "false negatives," which are attacks gone unnoticed by the IDS, both of these lead to a viable system's resources being wasted. The amount of false positives clogs up log files with erroneous reports, thus masking a legitimate attack in a sea of false alarms. Most systems administrators will ignore the IDS's data due to this fact. The other problem stems from attacks appearing to be friendly, normal traffic, which is even more alarming, since an attacker would be able to creep into the network without an alert from the IDS [8].

In this paper, we have evaluated the Mehran University Network for intrusion detection by benchmarking the data set and evaluation. Running the data through an intrusion detection system with a variety of configuration settings discovered these inconsistencies. The research also analyzes the actual way in which the test was taken and evaluated originally. This research will try to prove that the evaluation is not an acceptable way to measure the performance of an intrusion detection system, and may actually impede development of better systems due to evaluation based on a bad standard.

2. Literature Review

The 1998 evaluation consisted of seven weeks of test learning data. The purpose behind producing this was to give IDS evaluators a chance to tweak their rules based and anomaly detection systems by familiarizing them with the typical traffic running through the network. There were also attacks thrown into each of the learning data files, to show typical attacks. It gave the systems using datamining and learning algorithms a chance to have sample data, which helped them "learn" how the network operated [9].

A packet sniffer was located on the internal network to capture the generated traffic. All simulated attacks were launched from "outside" the base, so a traffic sniffer located at the gateway would be able to catch it all. Intrusion detection systems were supposed to detect the following categories: denial of service (DoS), port scanning and probes, user to root attacks (U2R), and remote to local (R2L). In the DoS categories, which should be fairly easy to detect, the best system could only pick up 65% of attacks. The probes category had two of the systems detecting 90% of probing activities [10].

The 1998 evaluation was only to provide exploits against UNIX hosts and was only supposed to initially be used for IDS that had been developed using DARPA grants . The oversight of Windows NTwhen it was arguably the most popular business operating system at the time of the evaluation, was probably used by government and military personal as well. Leaving this particular operating system out really harms the 1998 evaluation's credibility [10].

One of the major drawbacks in running the evaluation is that a listing of attacks in the actual test data is not available. In week 6 of the testing data, a bad router added too many ICMP packets into the data set, as computers constantly "pinged" each other. This generated massive log files and a major slowdown in Snort performance, as it logged over 1.5 million alerts upon completion, in some instances.

The two biggest problems, though, were the methodology used to come up with the exploits, and the way they were executed. There were 38 different kinds of malicious traffic used, but they were executed in no real logical order. Some sort of attacker intelligence should have been placed into the attack routines, even in the preliminary data.

The 1999 evaluation set out to improve upon the evaluation performed a year earlier, with extensions added on and more attack types. This included the addition of Windows NT exploits. The test bed to generate this particular data was similar to the 1998 tool, but also included updated statistics and the addition of Windows NT hosts. Also, the same attack sub-categories were used and are listed above [9].

This evaluation had many improvements over its predecessor, although it was still far from perfect. The scoring mechanism is similar to the one used in the 1998 evaluation and is, once again, a problem. I found the listings of the individual IDS performance to be confusing and the evaluation still did not use the criteria discussed in the section on the 1998 evaluation. A less confusing and more honest representation of the data would be to just list the amount of attacks found versus the amount of false positives. Capping it at a certain number and detection percentage leaves out the systems that did not "make the grade." The full listing is easier to represent and makes more sense [10].

3. Experimentation Model

3.1 Initial Model

The initial set-up was, with three computers connected with a hub (Figure 1). This process was as not easy as anticipated. The main problems that occurred stemmed from lack of time and resources.

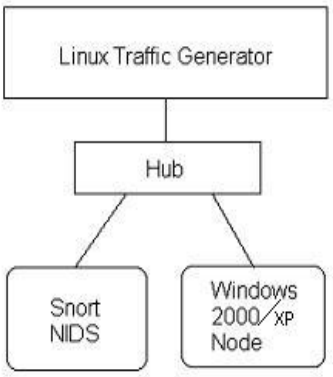

Fig. 1 Initial Set-Up

3.2 Configuring Snort

Snort is an open-source project started by Martin Roesch and is available on a variety of platforms for download, including Windows, Linux, and Solaris. It has been around for a seven years and acquired a decent number of developers and users. From personal use, it seems to be quite easy to customize and very flexible in terms of possible uses.

Below is a selection of the configuration file. The comments in the file are very descriptive (comments are preceded with the '#' symbol) and there are a number of options defined, such as what network ports to watch and which side (client, server or both) of the connection.

3.3 Output after Configuration

Start Apache and then go to http://yourhost/acid/acid_main.php. You will get a message that looks like this in your browser

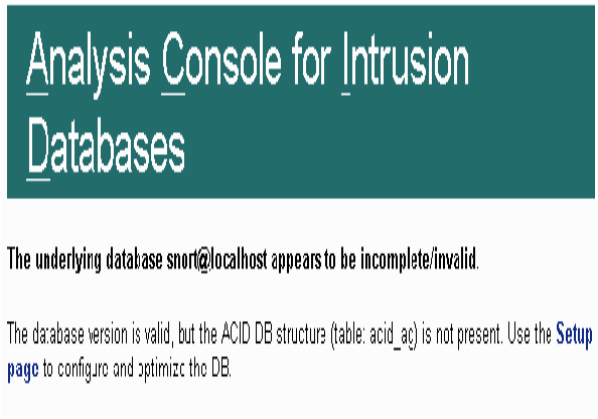

Fig. 2 Analysis Console Setup

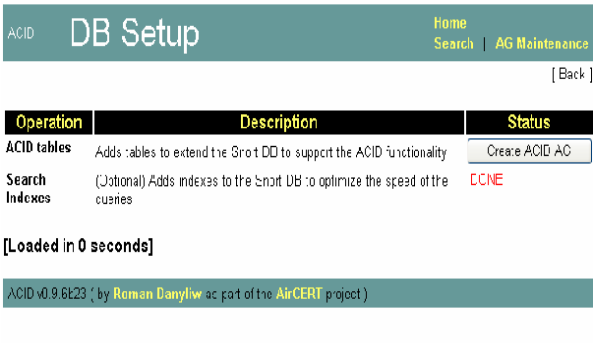

Fig. 3 Analysis control for intusion detection

Then click the button "Create Acid AG"

Now when you go to http://yourhost/acid/ you should see the ACID homepage

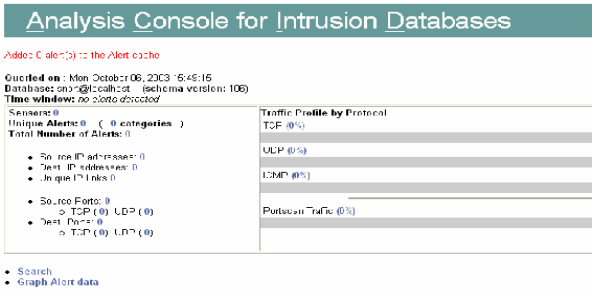

Fig. 4 Anlysis control for intrusion detection

3.4 Securing the Acid directory

mkdir /www/passwords
/www/bin/htpasswd-c /www/passwords/passwords

acidn(acid will be the username you will use to get into this directory, along with the password you choose)
It will ask you to enter the password you want for this user, this is what you will have to type when you want to view your acid page
Edit the httpd.conf (/www/conf) and include the following under the section that starts
with </Directory>
<Directory "/www/htdocs/acid">
AuthType Basic
AuthName "SnortIDS"
AuthUserFile /www/passwords/passwords
Require user acid
</Directory>
Now restart the http service (/etc/init.d/httpd restart) and next time you go to the acid
Webpage you will get a prompt for a username and password.

3.5 Check to see if everything is working:

Reboot your system; watch to make sure everything starts. You can check by doing a "ps –ef |grep <service>" the service can be any running process. i.e. mysql, httpd, snort, etc.

If you want the machine to start at a text prompt instead of X, then change the default in the inittab file (/etc/inittab) from 5 to 3. Go to a shell as root and check everything important to see if it is running.

To check you can execute "ps –ef |grep <SERVICE>" where service is snort. httpd, or mysql.

Or use "ps –ef |grep httpd && ps –ef |grep mysql && ps –ef |grep snort"

4. Proposed Solution and Implementation for the MUET Network

The current Network of MUET is shown below

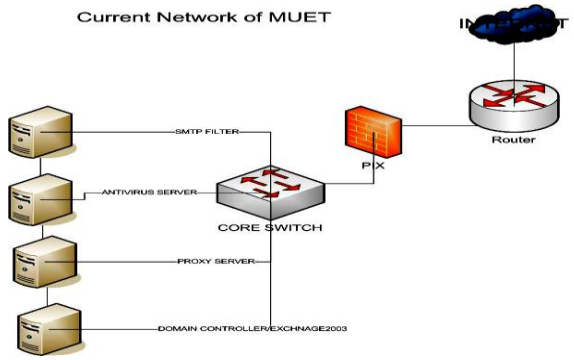

Fig. 5 Current MUET Network

4.1 Suggested Network for MUET

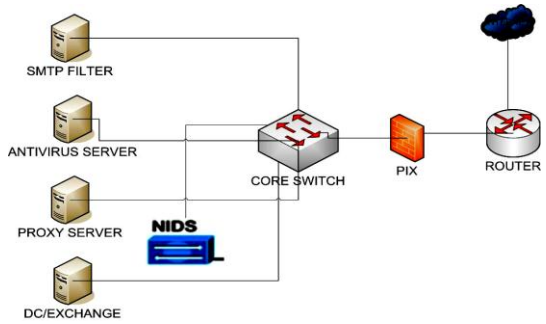

Fig. 6 suggested Network

4.2 Results & Discussion

Currently pix firewall is used for the security of MUET network and Symantec web security is using for the content filtering. The function of Firewall is to block all outbound and inbound port but not to send any alert if some hacker or cracker is entering in to the system. According to this scenario Network Intrusion Detection System (NIDS) should be connected with the core switch because every request in the network passes from the core switch so if there is anything going wrong then it will be detect by Intrusion Detection System (IDS) with the help of the rules which are described in the snort rules directory and also do the content filtering which is the extra feature of this IDS.

5. Vulnerabilities Found

The vulnerabilities found after running snort with the integration of ACID console are shown below.

These are the graphical output of the scanning results scan by snort.

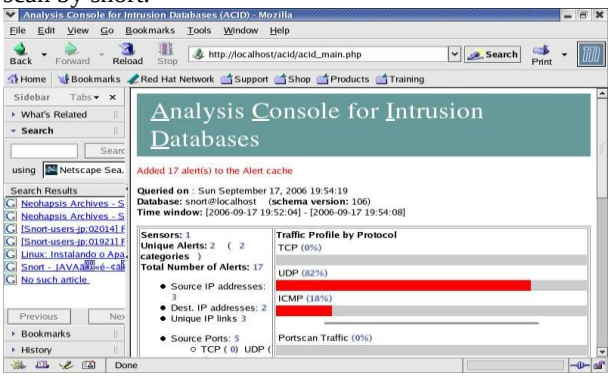

Fig. 7 Snort output using Acid

5.1 Unique IP Alert

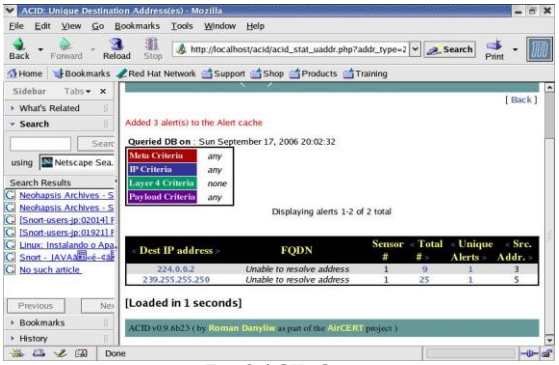

Fig. 8 ACID Output

5.2 Most frequent Alert

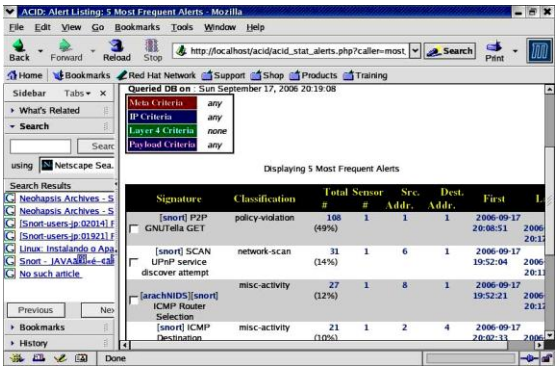

Fig. 9 Most Frequent Alerts

5.3 Alerts by IP

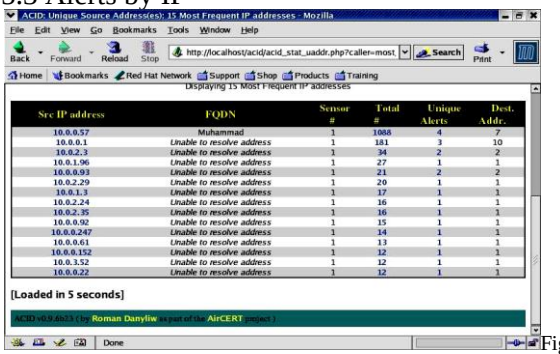

Fig. 10 Displaying Alerts by IP

Mostly the attacks found in the experimental scenario are Denial of service (Dos) attacks. The most common attack found is buffer over flow attack. This kind of vulnerability is perfect for remote access attacks because it gives the attacker a great opportunity to launch and execute their attack code on the target computer. A buffer overflow attack occurs when the attacker
intentionally enters more data than a program was written to handle. The data runs over and overflows the section of memory that was set aside to accept it.

5.4 Rule for Preventing Vulnerability

The rule written for the prevention of this vulnerability is shown below

activate tcp !HOME_NET any ->$HOME_NET 143 (flags: PA; \
content: "|E8C0FFFFFF| /bin"; activates: 1; \ msg: "IMAP buffer overflow!";)
dynamic tcp ! $HOME_NET any -> $HOME_NET 143 (activated by: 1; count: 50 ;)

The benefit of this scenario is that after the implementation of Intrusion Detection system in the MUET network the network become smart enough to handle intruder attacks and send alert if some hacker wants to hack the network. It can also be able to do the content filtering such as abuse words and porn websites. The main benefit of this IDS is that it's free and the Rule updates are easily available on SNORT web site.

6. Conclusion & Future Work

The experimentation of this project revealed how easy it was to edit Snort's rule sets based on attack listings in the 1998 evaluation and 1999 test data. It was possible to drastically reduce the number of false positives in the 1999 test data while still being able to detect most, if not all, attacks that the full rule set could detect.

This paper points out the flaws of the DARPA-LL Intrusion Detection System Evaluation in order to be able to improve upon them and develop a better method of testing, if, indeed, one could be found. It is my opinion, based on my investigation of this benchmark, that it falls short of its intended goals, although it is the best system devised to date. To build an all-inclusive intrusion detection benchmarking system would be a monumental task, but not necessarily an unreasonable one. Several factors would need to be considered in order to do this properly. The system would have to evolve as network applications and systems evolve. Also, since these systems are used to protect a variety of networks, all with different needs and configurations, a system based on each networks average traffic should be used, with the attacks listings inserted.

Designing an easily updated, intelligent evaluation system for IDS would be quite useful, but data sets would vary from company to company, as the usual traffic of most corporate networks can vary greatly. Developing a test similar to the Lincoln Labs evaluation would not be very useful. Further advances in data mining and artificial intelligence could help devise better evaluations.

Also, this paper does not mean to suggest that testing individual systems is too complicated. Anyone who uses an intrusion detection system should constantly test and configure it for the network they are trying to protect. Another topic of further research would be to establish a set of guidelines that systems administrators could use to devise their own tests.

References

[1] Lawrence Livermore National Laboratory Sandia National Laboratories December, 1996 at http://all.net/journal/ntb/ids.html

[2] Durst, Robert et al. "Testing and Evaluating Computer Intrusion Detection Systems". Communications of the ACM Volume 42 Issue 7 (July 1999): 53-61

[3] Fawcett, Tom and Foster Provost. Activity Monitoring: Noticing interesting changes in behavior. Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD-99). San Diego, CA, 1999. New York: ACM Press, 1999.

[4] http://www.snort.org.

[5] Roesch, Martin. "Snort – Lightweight Intrusion Detection for Networks". http://www.snort.org/docs/lisapaper.txt. 1999.

[6]SANS Institute. "101 Security Solutions". http://www.101com.com/solutions/security/article.asp?articleid=569. 2001.

[7] Hoagland, James A., and Stuart Staniford. Viewing IDS alerts: Lessons from SnortSnarf. Proceedings of 2001 DARPA Information Survivability Conference and Exposition (DISCEX 2001). Anaheim, CA, 12-14 June 2001. New York: Institute of Electrical and Electronics Engineers, 2001.

[8] Stonesoft at http://www.stonesoft.com/export/download/pdf/wp_IPS_winning_the_battle_A4.pdf.

[9] Lippmann, Richard et al. Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. Proceedings of DARPA Information Survivability Conference & Exposition (DISCEX), Hilton Head, South Carolina, 25-27 January 2000. Los Alamitos, CA: IEEE Computer Society, 1999: Vol. 2, 12-26.

[10] Lippmann, Richard et al. "The 1999 DARPA Off-line Intrusion Detection Evaluation". http://www.ll.mit.edu/IST/ideval/pubs/2000/1999Eval-ComputerNetworks2000.pdf, 2000.

Nash, David A., and Ragsdale, Daniel J. Simulation of self-similarity in network utilization patterns as a precursor to automated testing of intrusion detection systems. Proceedings of the 1st Annual IEEE Systems, Man, and Cybernetics Information Assurance Workshop, West Point, NY, June 6-7, 2000, pp. 53-57

[11] Barruffi, Rosy, Michela Milano, and Rebecca Montanari. "Planning for Security Management". IEEE Intelligent Systems & Their Applications. Los Alamitos, CA: IEEE Computer Society, Publications Office.

[12] http://rhn.redhat.com/help/latest-up2date.pxt

[13] Classless Inter-Domain Routing or CIDR. RFC 1519 at http://www.rfc-editor. org/rfc/rfc1519.txt

[14] The nmap at it web site http://www.nmap.org

[15] Open NMS at http://www.opennms.org

[16] The arachnids web site at http://www.whitehats.com/info/IDS

[17] The security focus mailing list archive at http://online.securityfocus.com/archive/1

Muhammad Ali Received his Master's in Communication Systems & Networks from Mehran University of Engineering & Technology, Sindh, Pakistan in 2007. He has been working as a System Engineer in information communication department of Mehran University from 2004 to 2009 and is currently working as a Lecturer in Northern Borders University Saudi Arabia in the Department of Computer Science. His research interests include Network securities, Wireless Networks, & Cloud Computing.

Jamshed Mustafa received the Master degree in Electrical Engineer from University Of Detroit Mercy, Detroit Michigan USA in 1991. He is working in Northern Border University Saudi Arabia in department of Electrical Engineering as Lecturer. His research interests include Computer networks, Network security and Wireless networks.

Mahera Erum Baloch received her Master's degree in Communication Systems & Networks from Mehran University of Engineering & Technology, Jamshoro, Pakistan, in 2011. She is currently studying towards a PhD degree in Computer Engineering at the University of Duisburg-Essen, Germany. Her research interests include Distributed & Parallel Computing, Performance Analysis & Evaluation techniques, Computer Supported Collaborative Work, & Artificial Intelligence.