Journal of the Nigerian Association of Mathematical Physics
Volume27, (*July, 2014*), pp 479 – 484

Mitigating Network Attacks Using Snort IDS

Konyeha Susan and Onibere Emmanuel

Department of Computer Science, University of Benin.

Abstract

Security is a big issue for all networks in today's enterprise environment. Hackers and intruders have made many successful attempts to compromise high-profile company networks and web services. Many methods have been developed to secure the network infrastructure and communication over the Internet, among them the use of firewalls, encryption, and virtual private networks.

In this work, a new detection method based on object oriented programming using Snort (an open source network intrusion detection system (NIDS)) has been configured and applied to a network to provide an additional layer of defense. This novel method monitors network traffic for predefined suspicious activity patterns and alerts system administrators when potential hostile traffic is detected. The development methodology is based on signature recognition. A combination of software packages have been used in the system development for enhancing the IDS usage. With these powerful technologies, the system is not only expected to be workable, but also highly efficient in terms of execution speed and response time. The system was tested and shown to be useful for identifying attacks and suspicious activity from examination of logged data, which are also presented in graphical form.

Keywords: Network intrusion detection, signature based detection.

1.0 Introduction

With the increase of communications, economy, industry and business dependence on the information technologies, the risk related to the pervasive intrusions in the electronic space is also increasing. Malicious intruders more frequently overcome protection systems installed in banks or companies intended to restrict the access to the computer network resources of the organization. Seeking to reduce the risk and possible consequences it is very important to identify the intrusions at the initial stage of their realization and to react to them properly [1, 2, 3].

Intrusion Detection System (IDS) is a relatively new addition to the field of computer security. It is concerned with software that can distinguish between legitimate users and malicious users of a computer and make a controlled response when an attack is detected. This paper mainly focuses on using NIDS for detecting network vulnerabilities and threats and thus providing an extra layer of security to computer systems, where the company or organization is currently using antivirus software, firewall and policies for security of its computer systems.

There are a lot of programs on computers that will want to open up network connections. Some of these programs have valid reasons for connecting, others have been written by people with motives ranging from questionable to criminal. To protect computers, mechanisms must be implemented to detect network access, and identify the source address and content.

An intrusion is an attempt to break into or misuse a computer system. The word "misuse" is broad, and can reflect something as severe as stealing confidential data or something minor such as misusing email system for spam. One of the most wide–spread network incident forms is the network scanning, which is used by the attacker for the configuration determination of the target network. For instance, the attacker can be interested in the active network hosts (servers, operating computers, etc.) and their services (web, e-mail, file sharing, etc.). An IDS is able to detect such intrusions. A network intrusion detection system (NIDS) monitors packets on the network grid and attempts to discover if a hacker/cracker is attempting to break into a system (or cause a denial of service attack). A typical example is a system that watches for large number of TCP connection requests (SYN) to many different ports on a target machine, thus discovering a TCP port scan attempt. A NIDS may run either on the target machine which watches its own traffic (usually integrated with the stack and services themselves), or on an independent machine promiscuously watching all network traffic (hub, router, probe). Thus a "network" IDS monitors many machines, whereas the host IDS monitors only a single machine (the one they are installed on).

Corresponding author: Konyeha Susan, E-mail: susan.konyeha@gmail.com, Tel.: +2348060826547

Bot is short for software robot, and is referred to as compromised PC attached to the internet and remotely controlled by a hacker. Some estimate that 25% of all broadband PCs are infected by bots, and that there are over a million bots available to participate in different types of attacks [4]. Hackers use communication systems, typically the internet chat application, to control the bots. The malicious code can get onto the PC through an e-mail attachment, 'silver wrapped' in a file which is automatically installed when visiting a web site, or in an mp3 file carried in a peer to peer application, to name a few common ways Trojans propagate. Once the malicious code executes, the bot will install itself, may patch the system, open service ports on the machine, and spread itself further on to other machines that it can reach from inside the network. The bot then sets up a connection to a Herder, the server in control of a number of bots, a botnet. It may be very difficult to detect that a PC has turned into a bot. In fact, it can even be hard to find out that it is communicating at all. This makes bot mitigation very challenging. Botnets can be a huge network of as much as 400,000 infected computers [5]. These armies of compromised PCs serve two main purposes – to launch spam e-mails for scam marketing, or to launch DDoS attacks. Bots are also used to send phishing e-mails, upload adware, and as key loggers to trace credit card information, passwords, or other personal information.

2.0 Signature Based IDS

A signature can be described as a conditional rule, which is tested on an instance of activity, identifying a specific type [5]. Signature based intrusion detection systems rely on a set of rules (also known as signatures) for detecting intrusive activity. The detection is done by comparing data packets with the monitored signatures or attributes against a database of known intrusions to decide whether the observed traffic is malicious or not. Detection alerts are generated based on specific attack signatures. These attack signatures encompass specific traffic or activity that is based on known intrusive activity.

Pattern matching is one of the simplest forms of signature or misuse detection in which the IDS search an event stream (log entries or network traffic) for occurrence of specific patterns/signatures.

Benefits of intrusion detection using signature schemes include:

i. Signature based detectors are very effective at detecting attacks without generating an overwhelming number of false alerts.

ii. Signature based detectors can quickly and reliably diagnose the use of a specific attack tool or technique. This can help security managers prioritize corrective measures and track security problems on their systems.

2.1 Snort

Snort is an open source signature Intrusion detection system developed by Martin Roesch and Chris Green [6]. It has the ability to parse network traffic and perform real-time traffic analysis and packet logging on networks [7]. Snort performs protocol analysis, content searching, and content matching. The program can also be used to detect probes or attacks, including, but not limited to, operating system fingerprinting attempts, common gateway interface, buffer overflows, server message block probes, and stealth port scans. Snort can be combined with other software such as Snort Snarf, sguil, OSSIM, and the Basic Analysis and Security Engine (BASE) to provide a visual representation of intrusion data for easy analysis of outputs. It has the most numerous and active community in the open source NIDS field today. Schema of Snort framework is presented in Figure 1

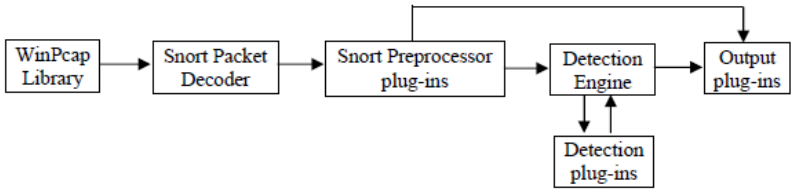

Figure 1: Schema of the Snort Framework [6]

The schema of the Snort framework shows the packet processing flow. The component layers in the Snort Framework function as follows:

WinPCap: Snort receives all the traffic by using the packet capture library. In Windows Systems this is done by the WinPcap library which is used to get access to Ethernet packets. First, Snort receives unprocessed packets from the Data Link Layer.

Packet decoder: This component processes the header information and decodes the data link frame, which can be for example 802.3 (Ethernet) or 802.11 (Wireless LAN). Thereafter, the IP protocol header and the TCP or UDP packet header will be decoded. Snort's decoder fills up internal data structures with the information provided by the packets and protocols. By this it enables the start of the inspection at the next level. The data structures are accessible from all higher layers and provide the needed information for further processing.

Preprocessor module: This module includes several plug–ins and operates on the data structures of the packet decoder. It implements various detection mechanisms which inspect the packets on this low level. In the preprocessor module also data transformations are performed which should simplify the further data inspection. Preprocessors can directly access the output modules and they are able to alert on suspect packets immediately. They also can classify the packets and possibly drop them if one of the preprocessors has identified an attack.
Detection Engine: The detection engine executes the main pattern inspection techniques. Here the packets passed on from previous layers are inspected at the transport and application layers, which are the layers 3 and 4 in the TCP/IP model. The packet contents are examined by rule-based detection plug-ins using signatures of already known attacks.
Output Module: This module generates and logs an alert if one of the previous layers has identified significant patterns in a packet. The detection engine and the preprocessors are directly connected with the output module. If a preprocessor or a rule of the detection engine responds on the contents of a packet then the output module receives a description of the incident. In the generated alert also some optional information about the detection activity can be provided.
Snort's strength are that it is free, open source, it's detectors are developed using tcpdump like syntax, powerful yet simple and very flexible, can run on a variety of platforms, output is in text file format, and is well supported in the open source community. Because of Snort's popularity there is a great deal of support and documentation available for Snort. Bleeding Edge Snort is a community which releases signatures and general rules that are able to detect attacks, as well as Sourcefire who are the original developers of Snort.
The base engine is a plug-in module which provides an added functionality for analysis of Snort output logs and alerts.

3.0 Methodology

We implemented architecture of signature type NIDS using Snort and performed signature detection using Snort rules. We generated network traffic from a Local Area Network which was set up consisting of a server (with Internet access), ten clients and a switch. Clients on this network were assigned IP addresses: 192.168.0.0/24 and 192.168.1.0/24. The signature detector was tested with live traffic from the Local Area Network setup and then with a standard dataset (1999 IDEVAL DARPA dataset). We triggered alerts by performing activities on hosts on our LAN that would violate Snort rules and hence trigger alerts. Figure 2 shows the architecture of Signature type Network Intrusion Detection System based on Snort

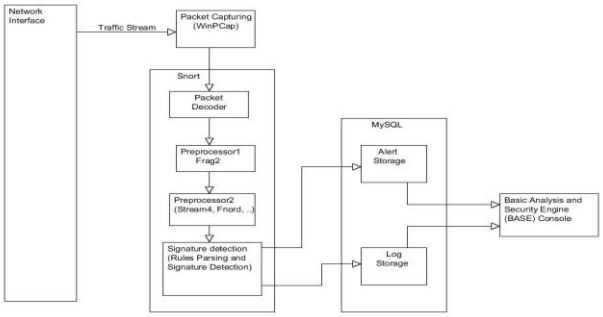

Figure 2: Architecture of signature type NIDS model using Snort [7]

The normal activity in a cyber café consists of accessing e-mails, browsing, authentication, uploading and downloading of files etc requiring the following network protocols: tcp, http, ack, arp, dns, https, and arp etc. A log or record of these packet traces was made and used for our experiments. We also downloaded 1999 IDEVAL DARPA dataset, which have both attack data and attack free data, for our experiments. The set up of the test bed for data collection and testing is shown in Figure 3.

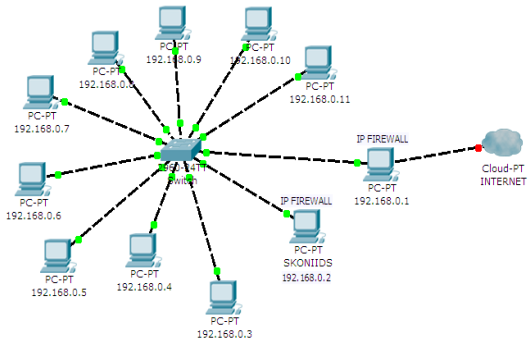

Figure 3: Set up of test bed

3.1 Procedures for Signature Detection

Procedures for Signature or Rule based detection

Step 1: We downloaded Snort signatures from www.bleedingsnort.com and copied the rules into Snort rules folder.
Step 2: We included it into the snort config file
Step 3: We configured the network variables in the Snort config file.
Step 4: Since we are using a logfile of the data collected. We executed at the command prompt, the command:
C:\example\anomalydetector>java -jar anomalydetector-3.0.0.SKONI.jar test logfile
The system begins to process the packet and match it against the rule. It checks for any offending packets. Once there is a match, the system provides an alarm response. If no alarm is raised, the data is simply logged into the database. It is more convenient to analyse the logs and alerts using a web based interface known as Basic Analysis and Security Engine (BASE). Hence Snort was configured to log its output to MySQL database for later analysis using BASE

4.0 Results and Discussions

A total of 49 TCP connections were logged, each of which is a data path triplet consisting of 10 source IP address and 13 destination IP addresses. The network protocols were (TCP=49, UDP=771, ICMP<1%, 820 source port and 61 destination port. A self set, consisting of 34 unique IP Links and port combinations were extracted from the set. We screen captured the result of logging this data in the web based interface made possible by BASE as shown in Figure 4.

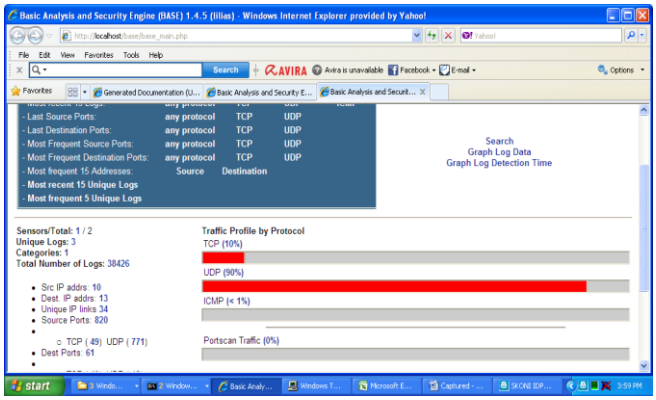

Figure 4: 38,426 logs from our LAN

4.1 Results for signature based detection Experiment using our LAN

We triggered alerts by performing activities on hosts on our LAN that would violate Snort rules and hence trigger alerts. We recorded 4 alerts for log displayed in the web based interfaces in Figure 5.

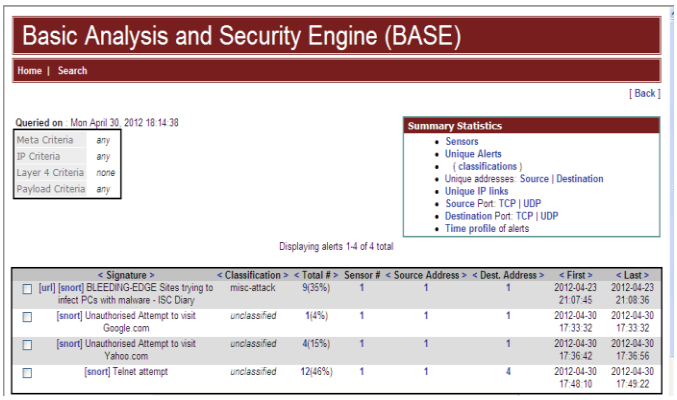

Figure 5: Alerts for performing unauthourised activities on hosts

We were able to detect 9 malware attacks on our LAN for the log file analyzed. The web based interface is shown in Figure 6.

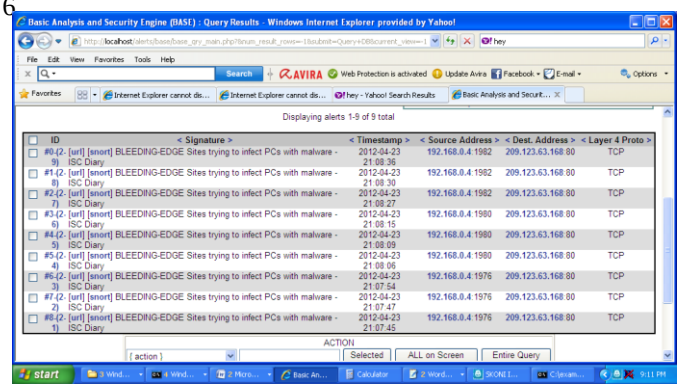

Figure 6: Nine (9) alerts generated by Malware attacks

4.2 Results for signature based detection Experiment using 1999 DARPA dataset

We downloaded the 1999 DARPA dataset for week 1, 2, 3, 4 and 5.to perform signature based detection. Below are the snapshots of alerts based on detected intrusions displayed using BASE in Figure 7 and 8 using IDEVAL DARPA dataset. Figure 9 is a bar chart for the 9 alerts obtained for signature detection using IDEVAL DARPA dataset

The alerts shown include nine different intrusive incidents, which were faithful logs of real incidents that occurred on the network being studied. Most of these attacks consisted of probing of one sort or another, particularly of services with recently reported vulnerabilities. At least one incident involved compromise of an internal computer.

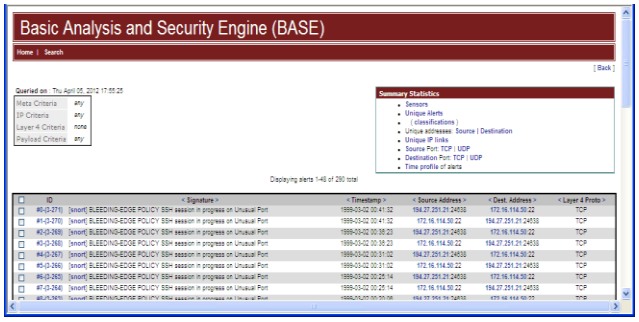

Figure 7: 1 to 48 of 290 alerts triggered for signature based detection using IDEVAL DARPA dataset.

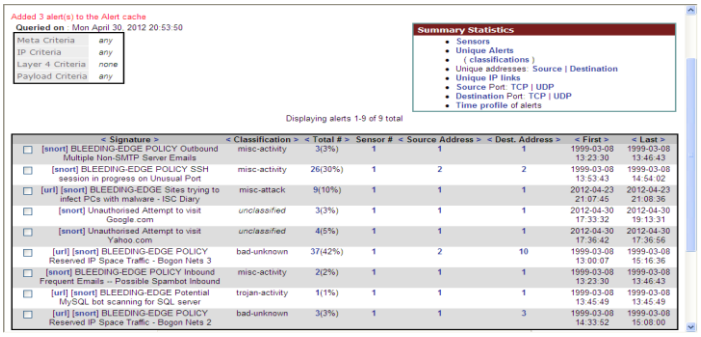

Figure 8: 9 alerts from signature detection using IDEVAL DARPA database

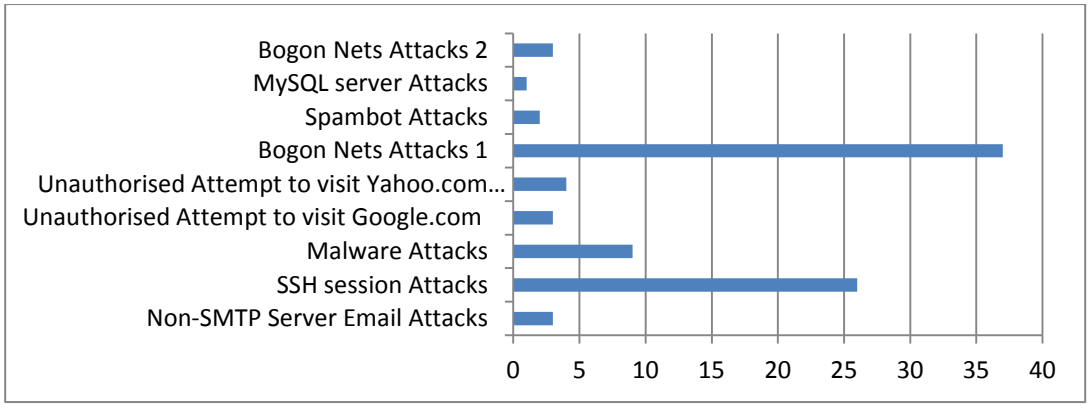

Figure 9: Bar chart of 9 alerts obtained for signature detection using IDEVAL DARPA dataset.

5.0 Findings

In testing for signature based attacks using Snort rules the intrusion detection system was able to detect eighteen(18) malware attacks, three(3) bogon nets attacks, a miscellaneous-attack, a telnet attempt, mysql bot scanning, spambot, SSH, multiple non SMTP server emails, an unauthorized attempt to visit goggle, an unauthorized attempt to visit yahoo.com which were all violation of policies based on the rules set and two hundred and ninety (290) TCP/IP alerts from violation of TCP signatures.

6.0 Summary and Conclusion

More and more intelligent and creative people are finding new ways to attack computer systems every day. Until recently, system administrators were limited to choosing from a small variety of protection mechanisms, including the useful, yet limited, firewalls and anti-virus solutions. Our experiments with Snort show that it is in fact a useful tool, especially when considering the speed with which rules can be released to protect against newly discovered attacks. It also adds the power of preventing attacks by looking at the application-layer information within the packet. This powerful combination will certainly serve the industry well over the next several years.

7.0 References

[1] Garsva E. and Skudutis J. (2004): Secure Computer System Design. Electronics and Electrical Engineering, No. 6 (55) pp. 43–48.

[2] Paulauskas N. and Skudutis J. (2008): Investigation of the Intrusion Detection System "Snort" Performance. Electronics and Electrical Engineering, No. 7 (87) pp. 15–18.

[3] Garsva E. (2006): Computer System Survivability Modelling by Using Stochastic Activity Network. Proceedings of SAFECOMP'06, Springer–Verlag. pp. 71–78.

[4] Livadas, C., Walsh, R., Lapsley, D., and Strayer, W. T. (2006): Using machine learning techniques to identify botnet traffic. In Proceedings of the 2nd IEEE LCN Workshop on Network Security (WoNS'2006).

[5] Zhuang, L., Dunagan, J., Simon, D. R., Wang, H. J., Osipkov, I., Hulten, G., and Tygar, J. (2008): Characterizing botnets from email spam records. In Proceedings of the First USENIX Workshop on Large – Scale Exploits and Emergent Threats (LEET'08).

[6] Roesch, M. and Green, C. (2005): Snort Users Manual 2.3.2. Snort. Documentation Sourcefire, Inc.

[7] Rehman, R. U. (2003): Intrusion Detection systems with Snort, Prentice HallPTR, New Jersey. p.12