The background traffic was generated using a Adtech AX/4000 broadband test system and a Smartbits SMB6000. The advantage of these traffic generators is that they are capable of generating sufficient traffic to saturate the network.

As for the results, almost all the network-based products tested showed good resilience to the IDS evasion technique. Despite the use of much horrified *fragroute*, which is an advanced hacking tool, the IDS products tested showed strong resistance. Although some IDSs still exhibit some weaknesses to certain individual evasion techniques, there was no evidence of any product completely succumbing to such advanced hacking tools. The full results of this evaluation can be seen in the 'verdict' sections in [NSS 2001] for every IDS evaluated.

However, on the downside, the direction taken by some of the IDS vendors seem worrying. They eliminate alerts for events which they deem unimportant. An example would be the IDS that does not raise a SYN flood alert unless it sees half-open connections, no matter how heavy the flood.

3.1.2 The 1998 DARPA-LL Evaluations [Lippmann 2000]

An intrusion detection evaluation test bed was developed which generated normal traffic similar to that on a government site containing 100's of users on 1000's of hosts. More than 300 instances of 38 different automated attacks were launched against victim Unix hosts in seven weeks of training data and two weeks of test data. Six research groups participated in a blind evaluation and their results were analyzed.

The purpose behind producing the seven weeks of training data was to give IDS evaluators a chance to modify the rules base and anomaly detection systems by familiarizing them with the types of traffic running through the network. There were also

attacks included in each of the learning data files, to show typical attacks. It gave the systems using data mining and learning algorithms a chance to have sample data, which helped them "learn" how the network operated [Lippmann 2000].

Since it was difficult to take real Air Force network data and manage to remove sensitive information from it for evaluation purposes, the lab used custom software to generate traffic. It allowed Lincoln Labs to simulate the activities of hundreds of programmers, managers and secretaries, as well as a few hosts appear to be thousands of terminals. A packetsniffer was located on the internal network to capture the generated traffic. All simulated attacks were launched from "outside" the base, so a traffic sniffer was located outside the gateway would be able to catch it all [Lippmann 2000].

The intrusion detection systems were supposed to detect the following categories: Denial of Service (DoS), Probe, User to Root (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 Probe category had two of the systems detecting 90% of probing activities. In the U2R category, the best systems could only find 60% to 70% of attacks. In the R2L, which allows remote users to gain local access (in some cases root access), the best system could only find 35% of attacks. The systems that could interpret BSM audit data on Sun workstations could improve performance slightly [Lippmann 2000].

3.1.3 The 1999 DARPA-LL Evaluations [Lippmann 2000a]

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. It also included Windows NT hosts. Also, the same attack sub-categories were used and are listed above [Lippmann 2000].

The full results of this evaluation can be seen in the summary of the results that Lincoln Labs published on pages 14 - 18 of its summary [Lippmann 2000a]. The scoring method Lincoln Labs showed in their charts was the percentage of attacks found with below 10 false alarms of that attack instance per day, and also had a detection rate above 40%.

Another major change in 1999 was a focus on determining the ability of systems to detect new attacks without first training on instances of these attacks. The 1998 evaluation demonstrated that systems could not detect new attacks well. The new 1999 evaluation was designed to evaluate enhanced systems which can detect new attacks and to analyze why systems miss new attacks. Many new attacks were thus developed and only examples of a few of these were provided in training data.

3.2 *Selection of IDS tool for the evaluation*

Among the objectives of this research is to perform testing and evaluation on intrusion detection systems. As this research is much constrained by time, resource and manpower, its scale is comparatively less complex than the evaluations discussed above. In this smaller scale, testing multiple types of intrusion detection tools would be impractical and superficial. A more viable way to perform a simple yet thorough evaluation is by testing **four** different configurations of one type of tool. To this end, an IDS tool called Snort was selected.

3.2.1 Why Snort?

The main reason for exclusively testing on snort, from which its different configurations are derived, is because it comes with a utility called SnortSnarf that parses the alerts from the alerts file. Due to the very large tcpdump test data file (described in sections 4.2), the number of lines generated in the alerts file can vary from zero to hundreds of thousands. Reading the alerts file line by line is too labour intensive and time consuming. SnortSnarf provides a simple to read report format of all the attacks detected by Snort. Other publicly available IDS tools do not have this utility thus rendering the testing for such a tool impractical for a one-person research.

Moreover, Snort was selected because it is easily obtainable and customized. 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. Martin Roesch, in his paper entitled "Snort – Lightweight Intrusion Detection or 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 network traffic as well as outright attacks".

3.2.2 What is Snort?

Snort is a lightweight network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and much more. Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture. Snort has a real-

ime alerting capability as well, incorporating alerting mechanisms for syslog, a user specified file, a Unix socket, or WinPopup messages to Windows clients using Samba's smbclient.

3.2.3 Snort Architecture

The Snort architecture consists of three principal components. A simplified representation of these components is shown in Figure 3.1. They are described below:

Packet Decoder: the Snort packet decoder supports the Ethernet, SLIP and PPP mediums. The packet decoder performs all the work to prepare the data in an expedient manner for the detection engine.

Detection Engine: the detection engine is at the heart of Snort. It essentially is responsible for analyzing every packet based on the Snort rules that are loaded at runtime. The detection engine separates the Snort rules into what is referred to as a chain header and chain options. The common attributes such as source/destination IP address and ports identify the chain header. The chain options are defined by details such as the TCP flags, ICMP code types, specific type of content, payload size, etc. The detection engine recursively analyzes each and every packet based on the rules defined in the Snort rules file. The first rule that matches the decoded packet triggers the action specified in the rule definition. A packet that does not match any Snort rule set is simply discarded. Key components of the detection engine are the plugin modules such as the portscan module. Plugin modules enhance the functionality of Snort by adding to the analysis capability.

Logger/Alerter: logging and alerting are two separate subcomponents. Logging allows the logging of information collected by the packet decoder in human readable

or tcpdump format. Alerts can be configured to be sent to syslog, flat file, Unix sockets or a database. Optionally, the alerting may be turned off completely during testing or penetration studies. By default, all logs are written in the /var/log/Snort folder, and all of the alerts are written to the /var/log/Snort/alerts file.

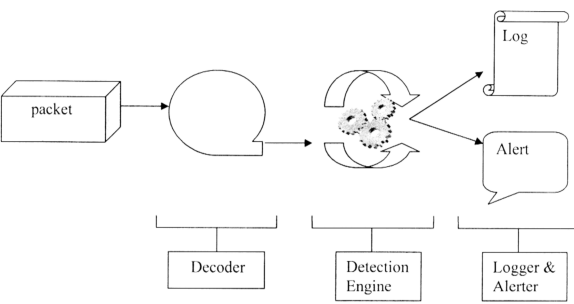

Figure 3.1: Snort Architecture

3.3 Test Objective

This research has three objectives. Firstly, is to perform a thorough evaluation process on intrusion detection system without the use of complex computing resources. Secondly, to evaluate the performance of the four configurations of the selected intrusion detection system, Snort. Finally, to uncover the limitations of the three past evaluations, mentioned in sections 1.2 and 3.1, after experience of IDS testing and evaluating is garnered.

3.4 Test Scope

The test was set up in accordance to the requirements detailed in section 3.5. This testing was implemented on Linux because the tcpdump test data sets can only be processed in a Unix-based environment.

The four configurations of Snort used in the testing were:

- Configuration 1 - Snort 1.7 Full
- Configuration 2 - Snort 1.7 Custom
- Configuration 3 - Snort 1.8.3 Full
- Configuration 4 - Snort 1.8.3 Custom

The different configurations were determined by the different ruleset used for the particular configuration. The four different ruleset configurations used are provided in Appendix E.

The four test data sets used (taken from: http://www.ll.mit.edu/IST/ideval/) for the testing was run through each of the above configurations. The test data sets are presented below:

Test Data Set 1 – 1998 Learning Data Week 6

Test Data Set 2 – 1998 Learning Data Week 7

Test Data Set 3 – 1999 Test Data Week 1

Test Data Set 4 – 1999 Test Data Week 2

These data sets were among the test data sets used in the DARPA-LL evaluation. The test data sets contain attacks and background traffic. The attacks can be divided into five categories, they are:

Denial of Service (DoS)

- Probe
- User to Root (U2R)
- Remote to Local (R2L)
- Data

The testing measures the performance of the four different configurations in terms of how well they detect the attacks in the test data. Then the result of the tests are put in report format by SnortSnarf (described in section 4.3.1 under subsection (E)) after which they are manually filtered and transferred to their Result Forms in Appendices F, G, H or I in the following order:

- Results from Test Data Set 1 – Result Form in Appendix F
- Results from Test Data Set 2 – Result Form in Appendix G
- Results from Test Data Set 3 – Result Form in Appendix H
- Results from Test Data Set 4 – Result Form in Appendix I

In total, there were 16 test runs - 4 test data sets were run through 4 configurations of Snort. As part of evaluating the IDSs, a performance ranking is conferred to each of the 4 configurations based on their overall performance as reflected in their test results.

3.5 Test Requirements

3.5.1 Hardware Requirements

The test machine must have at least the following minimum specifications:

- Intel Pentium III 500Mhz
- 96 MB of RAM
- 4 GB Hard Disk or greater

- All hardware must also be on the redhat hardware compatibility list.

  http://hardware.redhat.com/hcl/

3.5.2 Software Requirements

The following are the software required to support the running of the intrusion detection system:

Platform used for testing:

- Operating system - Linux 7.1 or later

Software Used for Snort:

- shadow-utils-20000902-12.i386.rpm - Utilities for managing accounts and shadow password files.

  http://rpmfind.net//linux/RPM/redhat/8.0/i386/shadow-utils-20000902-12.i386.html

- chkconfig-1.3.6-3.i386.rpm - A system tool for maintaining the /etc/rc*.d hierarchy.

  http://rpmfind.net//linux/RPM/redhat/8.0/i386/chkconfig-1.3.6-3.i386.html

- bash-2.05b-5.i386 - The GNU Bourne Again shell (bash) version 2.05b.

  http://rpmfind.net//linux/RPM/redhat/8.0/i386/bash-2.05b-5.i386.html

  glibc-2.2.93-5.i386.rpm – The GNU libc libraries

  http://rpmfind.net//linux/RPM/redhat/8.0/i386/glibc-2.2.93-5.i386.html

  glibc-debug-2.2.93-5.i386.rpm – Shared standard C libraries with debugging information

  http://rpmfind.net//linux/RPM/redhat/8.0/i386/glibc-debug-2.2.93-5.i386.html

  krb-libs-1.2.5-6.i386.rpm – The Shared libraries used by Kerberos 5

  http://rpmfind.net//linux/RPM/redhat/8.0/i386/krb5-libs-1.2.5-6.i386.html

- openssl-0.9.6b-29.i386.rpm – The OpenSSL toolkit
  http://rpmfind.net//linux/RPM/redhat/8.0/i386/openssl-0.9.6b-29.i686.html
- libpcap-0.6.2-16.i386.rpm – A system independent interface for user-level packet capture
  http://rpmfind.net//linux/RPM/redhat/8.0/i386/libpcap-0.6.2-16.i386.html
- postgresql-libs-7.2.2-1.i386.rpm – The shared libraries required for any PostgreSQL clients
  http://rpmfind.net//linux/RPM/redhat/8.0/i386/postgresql-libs-7.2.2-1.i386.html
- libstdc++-3.2-7.i386.rpm – GNU standard C++ library
  http://rpmfind.net//linux/RPM/redhat/8.0/i386/libstdc++-3.2-7.i386.html

3.6 Summary

This chapter discussed three previous evaluations: one from NSS Group, Europe's foremost independent network testing facility and consultancy organisation, and two from the MIT(Massachusetts Institute of Technology) Lincoln Laboratory, under Defense Advanced Research Projects Agency (DARPA) sponsorship. It then introduced the chosen IDS tool, Snort. The justification for the choice made is discussed in section 3.2.1 and an elaboration on the chosen IDS is presented in section 3.2.2. Following this, the test objectives, test scope and test requirements of this research are also discussed.