Global Information Assurance Certification Paper

Tutorial on Installing and configuring Snort on Fedora core 2: An Intrusion Analyst's, developer's & a researcher's perspective

GSEC Practical Version 1.4c Option 1

By Praveen D Ampatt

December 8, 2004

Abstract 3
Introduction 3
What is SNORT? 4
Part I Internals of Snort 5
Packet Decoder 6
Preprocessors 7
Detection Engine 10
Logging and Alerting Module 11
Output Modules 11
Snort Rules 12
Snort & Standards 14
Part II Configuring, Installing & Using Snort from an Intrusion Analyst's perspective 15
Before Installing Snort 15
Installation and Configuration of Snort on Fedora Core 2 with ACID as Front End and MySQL as Database 16
Snort and Reporting 21
Snort Usage 24
Snort Rule writing 25
Snort Testing 25
Snort & Some Useful Plugins 26
Securing Snort, the NIDS Itself 27
Snort kit 29
Tuning Snort 30
Part III Snort: A network IDS from a researcher's perspective 31
Evolving Trends in the field of Intrusion Detection 31
Making Compatible with IPV6 31
Enterprise IDS 31
Adaptive IDS 31
Immune IDS 32
Intrusion Prevention System (IPS) 32
IDS & IPS Scenario: Can Snort, IDS is still useful? 34
Standards and Consortium 34
References 35
Appendix 38

ABSTRACT

This paper carries out a study on one of the most popular Intrusion Detection Solution, **Snort,** a **Network IDS (NIDS).**

Part I of the paper discusses the inner technical details of Snort, its architecture and the underlying rule database and rule structure. It also covers functionalities, which could be enhanced and discusses this from a developer's perspective.

Part II of this paper serves as a tutorial to describe the installation, configuration and design of Snort. It covers performance-tuning measures & guidelines, which help in running and using Snort efficiently. One of the most important aspects of securing Snort itself has also been discussed. The paper discusses, Snort, more from the perspective of an Intrusion Analyst entrusted with the responsibility of configuring, installing and managing Snort.

Part III takes a wider perspective, aimed at a researcher who can visualize using Snort along with upcoming technologies, standards & varying environments.

INTRODUCTION

Most computer systems today within an organization exist mostly in a network environment. In such an environment there is a danger of it being misused/damaged or even used for illegal commercialization and cyber crimes. Countries around the world have started to recognize the danger and have started Computer Emergency Response Team (CERT) to tackle the problems and other dangers rising out of the above.

Such crimes are not always restricted to external networks but can occur within the internal network as well. This process of misconduct is called ***Network or System Intrusion.*** Its detection and safe guard has become one of the prime concerns for the professionals as well as Governments.

An ***Intrusion Detection System (IDS)*** is a device or application, which monitors computer systems and network traffic and analyse that data for possible attacks originating from outside the organization and also for system misuse or attacks originating from inside the organizations. IDS alerts security administrators, of suspicious activities that may be occurring on systems and networks in real time. Intrusion Detection Systems are classified into mainly two types: **Host based (HIDS)** and **Network based (NIDS).**

Host based Intrusion detection System or *HIDS* are installed as agents on a host (server). These intrusion detection systems looks into system and application log files to detect any intruder activity. Log Analysis, Audit trials, File Integrity checking and Vulnerability scanning are common methods used to find the attempted security breach.

Network intrusion detection System or *NIDS* resides as an agent on different critical points of the LAN in the form of sensors. It filters and analyses network packets in real time and compare them against a database of known attack signatures (known as *pattern matching*). The attack signatures are known methods that intruders have employed in the past to penetrate a network. Apart from the *pattern matching*, *Protocol Anomaly*

Detection and *Statistical based Anomaly detection* are the other two common method uses in the NIDS to find the attempted security breach. *Protocol Anomaly Detection* generally consists of decoding protocols all the way down the protocol stack and searches for the anomalies in the protocol, this could be non-compliance with the RFC standards and protocols specific exploits (HTTP URL encoding based attacks is an example of the protocol specific exploits). Statistical Anomaly Detection engine works on the principle that networks have well defined norms for traffic types and loads over a short to medium time frames. Profiles of the network's average activities are built and deviations from these norms are detected as "Anomalies", which could be a possible attack on the network.

Refer Appendix A for a comparison summary between HIDS & NIDS

What is SNORT?

Snort is an open source, cross-platform, software-based lightweight Network Intrusion Detection System (NIDS) developed by Martin Roesch of Sourcefire. Snort is capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, pattern 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 OS fingerprinting attempts. Snort uses a flexible rules language to describe traffic that it should collect or pass, and includes a detection engine utilizing a modular plug-in architecture. Snort has real-time alerting capability as well, incorporating alerting mechanisms for Syslog, user- specified files, a UNIX socket, or Win Popup messages to Windows clients using Samba's smbclient.

Suitable Plug-ins allows the detection and reporting subsystems to be extended. Available plug-ins includes statistical anomaly detection, database logging, small fragment detection, portscan detection, and HTTP URI normalization.

Snort can be configured to run in three modes. These are

- **Packet Sniffer**
  Snort's packet sniffing mode allows it to capture and display all network traffic to the administrator. It provides you with the flexibility to display either the entire packet or only certain header information.
- **Packet Logger**
  Snort's packet logging mode performs the same functionality as the packet sniffing mode but creates a traffic data file.
- **Network Intrusion Detection system**
  When ran in this mode, Snort is capable of detecting potential network intrusions using a rule-based intrusion-detection mechanism.

Snort & Supported Platforms[1]

Snort is supported on various platforms. The following platform matrix in Figure 1, taken from Snort website http://www.snort.org/about.html, can be referred before installing Snort.

[1] Snort website http://www.snort.org/about.html

i386	Sparc	M68k/PPC	Alpha	Other	
X	X	X	X	X	Linux
X	X	X			OpenBSD
X			X		FreeBSD
X		X			NetBSD
X	X				Solaris
	X				SunOS 4.1.X
				X	HP-UX
				X	AIX
				X	IRIX
			X		Tru64
		X			MacOS X Server
X					Win32 - (Win9x/NT/2000/XP)

Fig 1: Snort and its availability on various platforms

PART I INTERNALS OF SNORT

Snort being an open source software, and a very popular NIDS, it is a very helpful and a worthy learning experience for developers working in area of Network Security, Computer Networks & system software development and even Intrusion analysts who are entrusted with responsibility of deploying and using IDS. It is helpful for Intrusion Analysts, who, having knowing the internals can manage and use Snort more effectively.

An overall view of architecture of Snort is explained below with an emphasis on the working, feature and inner level details that have gone in developing Snort.

Architecture Of Snort[2]

Snort contains many configurable internal components that can vastly influence the performance of the snort. A good understanding of Snort's internal architecture is required to make Snort run and monitor for the intrusions effectively. Snort design incorporates five major components that are each critical to intrusion detection.

These components are: Packet Decoder, none, optional Pre-Processors, Detection Engine, Logging & Alerting System & various output modules. Figure 2, taken from [2] Rehman, refeeq, "*Intrusion Detection with SNORT: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID*" **http://www.informit.com/content/downloads/perens/0131407333.pdf**, depicts the components and how they are interfaced.

---

[2] Rehman, refeeq, "*Intrusion Detection with SNORT: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID*" http://www.informit.com/content/downloads/perens/0131407333.pdf

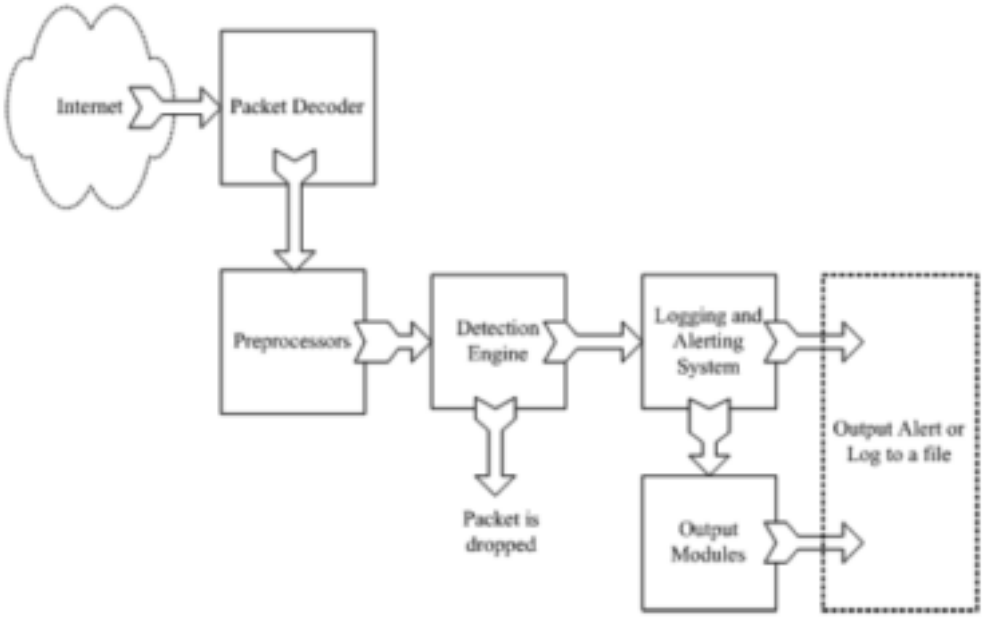


Fig 2, Architecture of Snort

A discussion of the above components is provided below.

Packet Decoder

The Snort's packet decoder decodes the packets in a layer wise approach and identifies the protocols/structures used at each layer. It is the first component of the Snort architecture. Snort decoder relies on an external packet-capturing library (libpcap[3]) to sniff packets from the network. Libpcap was chosen for packet capture for its platform independence. It can be run on every popular combination of hardware and OS. (Winpcap[4] is the windows version of the libpcap.). Libpcap captures the packet from the network in its original form as it is traversing across the network from the client to server. These packets will have all its protocol header information intact and unaltered by the operating system. Most of the network applications do not process the packets in its original format, they depend on the operating system to read protocol information and properly forward payload data to them. But Snort needs to have the packets in the original format because snort uses protocol header information in the packet, which would have been stripped off by the operating system to detect some forms of attacks.

Libpcap is not the most efficient way to acquire raw packets. It can process only one packet at a time, making it a bottleneck for high-bandwidth (1Gbps) Networks. There are several methods other than using libpcap for sniffing the packets from the networks. On Linux, a *modified version of*

[3] LibPcap Packet capture tool. Available at http://www.tcpdump.org/
[4] WinPcap Packet capture tool. Available at http://windump.polito.it/

libpcap[5] is available that implements a shared memory ring buffer. Berkeley Packet Filter[6] (BPF) and the SOCK_PACKET mechanism in the Linux kernel are other tools for grabbing raw packets.

Snort's packet decoder is actually a series of packet decoders that each decodes specific protocol elements. It works up the TCP/IP stack[7] staring with lower level Data Link Layer protocols, decoding each protocol as it moves up. Packet decoder decodes and stores the packet into a snort internal data structure. As soon as packet data is stored in the data structure it is ready to be analysed by the preprocessors and the detection engine.

Preprocessors

Preprocessors are written as "plug-ins" to allow them to give Snort a flexible extensibility and configurable on a host-by-host basis. Snort's preprocessors fall into two categories. They can be used to either examine packets for suspicious activities or modify the packets so that the detection engine can properly interpret them. A number of attacks cannot be detected by signature matching via the detection engine, so first type of preprocessors (which detect attacks) can be configured to detect those kinds of attacks. The second type preprocessors are responsible for normalizing traffic so that the detection engine can accurately match signatures. These preprocessors defeat attacks that attempt to evade Snort's detection engine by manipulating traffic patterns. An IP Fragmentation attack[8] is an example of this kind of attack. Additionally, Snort cycles packets through every configured preprocessor to discover attacks that require more than one preprocessor to detect them. If Snort simply stopped checking for the suspicious attributes of a packet after it had set off an alert via a preprocessor, attackers could use this deficiency to hide traffic from Snort. Suppose an attacker intentionally encoded a malicious remote exploit attack in a manner that would set off a low priority alert from a preprocessor. If processing is assumed to be finished at this point and the packet is no longer cycled through the preprocessors, the remote exploit attack would register only an encoding alert. The remote exploit would go unnoticed by Snort, obscuring the true nature of the traffic.
Preprocessor parameters are configured and tuned via the **snort.conf** file, which is discussed later in section (Snort Configuration). The details of the snort preprocessors and configuration parameters are available from Snort users manual[9].

Detailed View of Snort frag2 Preprocessor

The frag2 preprocessor is Snort's weapon against IP fragmentation attacks. Fragmentation is a normally occurring phenomenon in IP networks. The attack works by sending a fragmented TCP packet with header information that is allowed through the firewall. Subsequent fragments contain malicious data that would not otherwise be allowable through the firewall. A

---

[5] Modified version of the libpcap available at http://public.lanl.gov/cpw/
[6] . Berkeley Packet Filter Packet capture library available at http://www-nrg.ee.lbl.gov/
[7] RFC 1180 - TCP/IP tutorial http://www.faqs.org/rfcs/rfc1180.html
[8] RFC 1858 –Security Considerations for IP Fragment Filtering. http://www.faqs.org/rfcs/rfc1858.html
[9] Snort user Manual. http://www.snort.org/docs/snort_manual/node10.html

popular software, fragroute[10] can be used to craft fragmented packets with malicious contents.

frag2

There are five optional parameters that can be specified for the "frag2". These are:

- timeout <xx>
- memcap <xxx>
- detect_state_problems
- min_ttl <xx>
- ttl_limit <xx>

A short description of the above is provided below, some of the details referred from "Snort user Manual" available at http://www.snort.org/docs/snort_manual/node10.html

timeout *<seconds>*

This option sets the number of seconds that a fragment is to be saved. If the fragment is not completed in the defined time allowance, it is dropped

memcap *<bytes>*

The memory cap option limits the amount of memory the frag2 preprocessor can utilize

detect_state_problems

Turns on alerts for events such as overlapping fragments

min_ttl number

The min_ttl configuration option specifies the minimum time to live (TTL) that frag2 will accept. Anything lower than *[number]* will be dropped

ttl_limit *number*

This option specifies the maximum difference in TTL values that fragmented packets with the same fragment ID can have.

A brief view of the other available Snort Preprocessors is provided below

stream4

Snort uses stream4 preprocessor to provide tcp stream reassembly and stateful analysis capabilities to snort. Stateful inspection with `stream4` helps Snort better match attack signatures across multiple packets. This is important because attacks can be spread over many individual packets or exhibit anomalies in terms of TCP connections.

HTTP_inspect

The HTTP_inspect preprocessor is responsible for detecting abnormal HTTP traffic and normalizing it so that the detection engine can properly interpret it. HTTP_inspect works specifically with the URI string of an HTTP[11] request. It generates an alert if it encounters traffic that requires decoding. URL

[10]Hacker tool for creating malicious fragmented packets
http://www.monkey.org/~dugsong/fragroute/

[11] `Hypertext Transfer Protocol RFC` http://www.faqs.org/rfcs/rfc1945.html

Encoded Attacks[12] is one of the most common attack happens with the HTTP request.

Portscan detector

Portscan detector logs details of the portscan (start time, end time, source ip, destination ip, type of portscan etc) to the standard logging facility. A portscan is defined as tcp[13] connection attempts to more than p ports in t seconds or udp[14] packets sent to more than p ports in t seconds.

Portscan ignorehosts

Using this preprocessor, you can tell Snort to ignore tcp syn and udp portscans from certain hosts. The arguments to this module are a list of ips/cidr blocks to be ignored.

Flow

The flow tracking module is meant to start unifying the state keeping mechanisms of snort into a single place.

Flow-portscan

This module is designed to detect portscans based off flow creation in the flow preprocessors. The goal is to catch one to many hosts and one to many ports scans.

Telnet decode

This is another one of the family of decoding preprocessors. This preprocessor specifically relates to Telnet[15] and FTP[16] protocols. Telnet_decode decodes or removes arbitrarily inserted binary Telnet control codes in a Telnet or FTP stream.

RPC decode

The rpc_decode preprocessor normalizes rpc multiple fragmented records into a single un-fragmented record. it does this by normalizing the packet into the packet buffer.

Performance monitor

This preprocessor measures snort's real-time and theoretical maximum performance. Whenever this preprocessor is turned on it should have an output mode enabled, either "console" which prints statistics to the console window or "file" with a file name, where statistics get printed to the specified file name.

---

[12] Ollamann, Gunter "URL Encoded Attacks using the common web browser" Technical Info. http://www.technicalinfo.net/papers/URLEmbeddedAttacks.html

[13] Transmission control protocol(TCP) RFC http://www.faqs.org/rfcs/rfc793.html

[14] User Datagram Protocol (UDP)RFC http://www.faqs.org/rfcs/rfc768.html

[15] Telnet protocol Specification (RFC) http://www.faqs.org/rfcs/rfc854.html

[16] File transfer protocol (RFC) http://www.faqs.org/rfcs/rfc959.html

Detection Engine

The detection Engine is the most important part of the Snort. The responsibility of the detection engine is to search for any intrusion activities in a packet. The detection engine employs the Snort rules for this purpose. If a packet matches any rule, appropriate action is taken; otherwise the packet is dropped.

Detection Engine is the time critical part of the snort. Depending upon how powerful your machine is and how many rules you have defined, the system may takes different amounts of time to respond to different packets. If the network traffic in the network is too high and the no of rules deployed in the detection engine is also more, then there is a chance of the system dropping the packets.

The detection engine works in different ways for different versions of Snort. In all 1.X versions of Snort, the detection engine stops further processing of a packet when the rule is matched. Depending upon the rule, the detection engine takes appropriate action by logging the packet or generating an alert. This means that if a packet matches criteria defined in multiple rules, only the first rule is applied to the packet without looking for other matches. This can create a problem. A low priority rule generates a low priority alert, even if a high priority rule meriting a high priority alert is located later in the packet. In this case the more serious vulnerability is **not** getting noticed. This problem is rectified in Snort 2.X versions.

Snort 2.0 introduced a high performance Multi-Rule Inspection Engine responsible for detecting rule matches during packet processing. Packets are first analyzed to select the appropriate set of rules for the inspections. Then Multi-rule Inspection Engine searches for rule matches. In this scenario only a subset of rules has to be matched with the packet. This increases the performance of the snort. (Different tests shows that the Snort 2.X versions are eighteen times faster than Snort 1.X versions). Figure 3 depicts this increase in performance in a graphical manner. It has been taken from Roelker, Dan, "HTTP IDS Evasions Revisited " ,April 2004 available at http://www.sourcefire.com/products/downloads/secured/sf_snort20_detection_rvstd.pdf

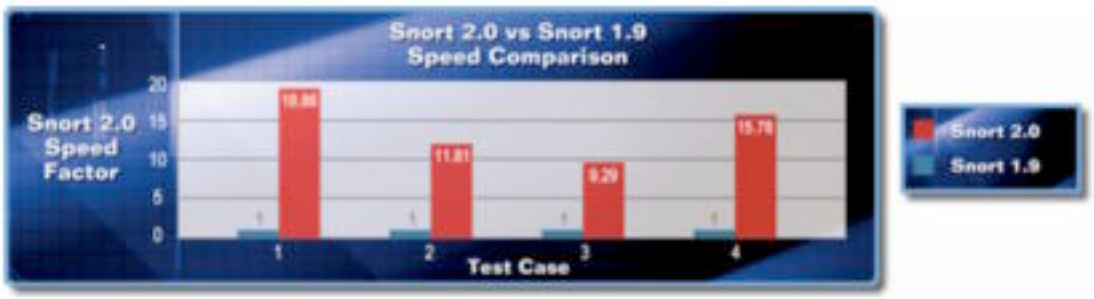


Fig 3: Snort comparison chart[17]

17 Roelker, Dan, "HTTP IDS Evasions Revisited " ,April 2004 http://www.sourcefire.com/products/downloads/secured/sf_snort20_detection_rvstd.pdf

Logging and Alerting Module

Depending upon what the detection engine finds inside a packet, the packet may be used to log the activity or generate an alert. Logs are kept in simple text files, tcp-dump style files or some other formats according to the configuration.

Logging and alerting systems send their data to **/var/log/snor**t by default or to a user directed directory (specified with the – l option in the command line).

Output Modules

Output modules or plug-ins can do different operations depending on how you want to save output generated by the logging and alerting system of the snort. Basically these modules control the type of output generated by the logging and alerting system.

The output modules will run when the alert or logging subsystems of snort are called, after the preprocessors and detection engine. Multiple logging and alert modules can be specified in the snort configuration file. When multiple plug-ins of the same type (log, alert) are specified, they are stacked and called in sequence when an event occurs output modules are loaded at runtime by specifying the output keyword in the rules file, name of the output module and the options corresponding to that in the following format:

output <name>: <options>

Various Output Modules Available For Snort

alert_syslog

This module sends alerts to the syslog facility (much like the -s command line switch). this module also allows the user to specify the logging facility and priority within the snort rules file, giving users greater flexibility in logging alerts.

alert_fast

Using this, Snort's alerts are printed in a quick one line format to a specified output file. It is a faster alerting method than full alerts because it doesn't need to print all of the packet headers to the output file

alert_full

This output module prints snort alert messages with full packet headers. The alerts will be written in the default logging directory (/var/log/snort) or in the logging directory specified at the command line. Inside the logging directory, a directory per ip address will be created. These files will be decoded packet dumps of the packets that triggered the alerts. The creation of these files slows snort down considerably. This output method is discouraged for all but the lightest traffic situations.

alert_unixsock

Sets up a Unix domain socket and sends alert reports to it. External programs/processes can listen in on this socket and receive snort alert and packet data in real time

log_tcpdump

The log_tcpdump module logs packets to a tcpdump-formatted file

database

This module sends snort data to a variety of sql databases

csv

The csv output plugin allows alert data to be written in a format easily importable to a database. The plugin requires 2 arguments, a full pathname to a file and the output formatting option.

unified

The unified output plugin is designed to be the fastest possible method of logging Snort events. The unified output plugin logs events in binary format, allowing another programs to handle complex logging mechanisms that would otherwise diminish the performance of Snort.

log null

This output modules is designed to alert the Snort events without logging that.

Details of the Snort internals were being referred from
"Snort Users Manual"
http://www.snort.org/docs/snort_manual/
and from Koziol , Jack's book "Intrusion Detection with Snort".

Snort Rules

There are several other details important from both an Intrusion Analyst and a developer. The following section discusses about the Snort's rule structure, how the rules are framed and the availability of rules.

Snort allows the user to write rules describing

- Well-known and common vulnerability exploitation attempts
- Violations of organization's security policy
- Conditions under which you think a network packet(s) might be anomalous

Most Snort rules are written in a single line. This was required in versions prior to 1.8. In current versions of Snort, rules may span multiple lines by adding a backslash (\ ) to the end of the line.

Snort Rules are logically divided into two parts, Rule Header and Rule Options.

Snort Rule Structure

Rule Header	Rule Options

The rule header contains the information about what action a rule takes. It also contains the criteria for matching the packets. Rule Header consists of five parts.

Rule Header

Action	Protocol	Source IP Address	Source Port	Direction	Destination IP Address	Destination Port

Action

It specifies what to do when a rule matches the criteria. There are five available default actions for snort. They are

- **alert** - generate an alert using the selected alert method, and then log the packet
- **log** - log the packet
- **pass** - ignore the packet
- **activate** - alert and then turn on another dynamic rule
- **dynamic** - remain idle until activated by an activate rule , then act as a log rule

Protocol

Snort currently analyses the following 4 protocols for suspicious behavior. The following are protocols supported by Snort

- **tcp**
- **udp**
- **icmp**
- **ip**

IP Address

Snort rules can be applied to a single host or a network. The addresses are formed by a straight numeric IP address and a netmask.
A netmask of
/24 indicates a class C network
/16 indicates a class B network.
/8 indicates a class A network
/32 indicates a single host.
Port Number

The port number specifies port or range of ports from which the traffic is originated or going to. Range of port numbers can be specified using colon(**:**). For eg. 2222:4444 indicates the ports from 2222 to 4444

Direction

The direction operator indicates the direction, of the traffic that the rule applies to.

-> Indicates the traffic from outside to inside
<- Indicates the traffic from inside to outside
<> Indicates the traffic in both direction

Rule Options

Rule Options Follows the rule header and enclosed inside a pair of parentheses. There can be one or many options and these options are separated with a Semicolon **(;)**. If multiple options are associated with a rule, then these options forms a logical AND. i.e. All the options has to be satisfied for the matching of the rule.
The above information on rule structue has been referred from Snort Users Manual http://www.snort.org/docs/snort_manual/node14.html
Rehman, refeeq, "Intrusion Detection with SNORT: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID" Chapter 3, working with snort rules
http://www.informit.com/content/downloads/perens/0131407333.pdf

Snort Rule Database

The rules come packed in the directory known as "rules" inside the main snort directory. The rules can be categorized based on the kinds of

attacks, application specific, protocols related and other related commonalities
The files in which rules are written are having ".rules" appended with their name.

Some example files named are

- backdoor.rules, describing about the backdoor rules
- mysql.rules, describing about the attacks on mysql database
- ftp.rules, describing about the attacks related to ftp protocol

Users can add their own rule files
All Snort Rules are available at Snort Rule Database:
http://www.snort.org/snort-db/
Also some sites frame their attack signatures according to Snort rule set. One of them is Whitehats website http://www.whitehats.com/ids/

Snort & Standards

Snort is CVE[18] compatible. The rules in the Snort's database are CVE adhere to the compatibility process specified here CVE Compatibility Process http://www.cve.mitre.org/compatible/process.html#declaration.

[18] Common Vulnerabilities & Exposures, CVE, http://www.cve.mitre.org/

PART II CONFIGURING, INSTALLING & USING SNORT FROM AN INTRUSION ANALYST'S PERSPECTIVE

An Intrusion Analyst that is going to use Snort, MUST know the details of Snort's configuration and various installation requirements and options. A lot of tools, applications and software are a pre-requisite to install Snort with a database logging and reporting environment which would help in managing the logs generated and also in analysisg the alerts and events reported.

Before Installing Snort

Before starting with the installation of Snort, the following precautions need to be taken

- Make sure that you have the approval of the Organization to run the Snort (preferably in writing).
- Decide on the network location to put the sensor. This is heavily influenced by the organization policies and what you want to detect. One way of looking at it is, determining if you want to place it inside the firewall or outside the firewall. Placing a sensor outside the firewall will allow you to monitor all packets directed to your network, regardless of whether or not they are stopped at the firewall. Place a sensor inside your firewall if you are only interested in monitoring traffic that your firewall let pass. If resources permit, it may be best to place one sensor outside and one sensor inside of your firewall.

  The following extract taken from Andrew R. Baker'sa article titled "Deploying Snort" taken from http://www.dpo.uab.edu/~andrewb/snort/deploying.html discusses the positioning of the Snort sensor and various issues.

  > In order for snort to be most effective, it needs to be positioned where it will see the most traffic possible. On a shared ethernet network, the detection interface can be simply connected to the shared network. In a switched environment, the detection interface will need to be connected to a "monitor port" on the switch. A "monitor port" is a port that is configured to receive all of the traffic that passes through the switch. The final, and possibly most popular, deployment is where the sensor is used to monitor all traffic coming (and leaving) an otherwise isolated network.

- Choose an Operating System to run Snort. Since Snort runs on almost all platforms, select an operating system, which suits best for your organization. Supportability, Stability, Performance and Security of the Operating System should be taken into account before selecting one.

The operating system's security is of paramount importance, as attacks keep on using the known and existing vulnerabilities and leverage on the unpatched systems. Outbreak of viruses and worms can dramatically

affect production units and business growth when assumptions regarding OS security are naively made.

IDS, or rather any security device best performs when the underlying OS is patched with latest security updates/patches.

Refer Section on “Steps to Secure the Snort NIDS machine” in this paper for more details

Installation and Configuration of Snort on Fedora Core 2 with ACID as Front End and MySQL as Database

This section discusses steps and instructions for installation and configuration of Snort on Fedora Core 2. Also ACID (Analysis Console for Intrusion Detection), a reporting tool that can be used to process Snort’s events is discussed.

Download all required Files

Download Snort
http://www.snort.org/dl/snort-2.2.0.tar.gz

Download LibPcap
Packet capture library.
http://www.tcpdump.org/release/ libpcap-0.8.3.tar.gz

Download PCRE
The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5.
ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-5.0.tar.gz

Download MySQL Source
MySQL Relational database
http://mysql.secsup.org/Downloads/MySQL-4.0/mysql-4.0.20.tar.gz

Download Apache HTTP Server Source
For the Webserver
http://httpd.apache.org/download.cgi/httpd-2.0.52.tar.gz

Download PHP
PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.
http://www.php.net/distributions/php-4.3.9.tar.gz

Download ADODB
ADODB is a database abstraction library for PHP
http://phplens.com/lens/dl/adodb453.tgz

Download ACID(Analysis Console for Intrusion Databases)

The Analysis Console for Intrusion Databases (ACID) is a PHP-based analysis engine to search and process a database of security events generated by various IDS, firewalls, and network monitoring tools
http://acidlab.sourceforge.net/acid-0.9.6b23.tar.gz

Download JPGraph
JpGraph is an OO class library for PHP >=4.3.0. JpGraph makes it easy to draw both "quick and dirty" graphs
http://www.aditus.nu/jpgraph/downloads/jpgraph-1.16.tar.gz

Download Zlib
This is a lossless data-compression library for use on virtually any computer hardware and operating system.
http://www.zlib.net/zlib-1.2.2.tar.gz

Installation and Configuration

Download all the files and store it in a folder. Then start installing applications followed by Snort. Some steps of the installation need the privileges of the *root*.

Required applications are

- PCRE
- Zlib:
- Libpcap
- MySQL
- Apache with PHP

On most of the Fedora Core 2 machines, the above softwares are installed and available by default. Appendix B discusses installation and configuration of the above softwares for OS, which does not have them pre installed

Steps for Installing and setting up Snort and the Snort rules

```
groupadd snort
useradd -g snort snort -s /dev/null
mkdir /etc/snort
mkdir /var/log/snort
mkdir /etc/snort/rules
Change directory (cd) to where you have downloaded and stored the files.
tar -xvzf snort-2.2.0.tar.gz
cd snort-2.2.0
./configure --with-mysql=/usr/local/mysql
make
make install
```

Snort Configuration

Snort uses a configuration file at the startup time. By Default **/etc/snort/snort.conf** is the configuration file. But you can specify any file

using the –c command line option. It is even possible to invoke multiple Snort instances on different network interfaces with different configuration.

This file contains six basic sections:

- Variable definitions
- Config Directives.
- Preprocessor configuration..
- Output module configuration.
- Defining new action types.
- Rules configuration and include files

Variable Definitions

Snort has many configuration variables and options, but the two most important ones are $HOME_NET and $EXTERNAL_NET. $HOME_NET is a variable that defines the network or networks you are trying to protect, while $EXTERNAL_NET is the external networks to which you are connected. These variables are used in virtually all rules to specify criteria for the source and destination of a packet.

e.g. var HOME_NET [172.16.5.0/16,172.16.55.0/16]
Above example specifies two networks which snort has to monitor,

Config Directives

Config directive allows a user to configure many general settings for snort. These directives can be used to replace many command line options as well.
Format of the Cofig parameter

```
config directive_name[: value]
```

Preprocessor Configuration

Using the this parameter various preprocessors can be configured .
General Format of this

```
preprocessor <preprocessor_name>[: <configuration_options>]
```

Output Module Configuration

Using the this parameter various Out Put Plugins can be configure can be configured

```
output <output_module_name>[: <configuration_options>]
```

Defining New Action Types

Using this this option user can specify new action types other than the predefined action types.

Rules Configurations and Include Files

Rules configuration is the place in the configuration file where user can put the rules, which has to be monitored for that particular Sensor . However the convention is to put all Snort rules in different text files. User can include these text files in the snort.conf file using the "**include**" keyword

Installing the rules and conf file:
(From the Snort installation directory)
cd rules
cp * /etc/snort/rules
cd ../etc
cp *.conf /etc/snort
cp *.config /etc/snort
cp *.map /etc/snort

Modify the snort.conf file:
The snort.conf file is located in /etc/snort, make the following changes.

var HOME_NET 10.2.2.0/24 (make this what ever your internal network is)
Change the rule path variable
var RULE_PATH /etc/snort/rules
Tell it to log to the database (make sure this is on one line) “new_password is what ever you want as long as it is the same when you set up mysql later)

output database: log, mysql, user=snort password=new_password dbname=snort host=localhost

Set snort to start automatically
Use the script located in the contrib. (cd ../contrib.) directory, S99snort. Copy it to /etc/init.d and call it snort. (cp S99snort /etc/init.d/snort) Change the following lines:
CONFIG=/etc/snort/snort.conf and
SNORT_GID=nogroup to snort
It should look like the following.

```
# Configuration
# set config file & path to snort executable
SNORT_PATH=/usr/local/bin
CONFIG=/etc/snort/snort.conf
# set interface
IFACE=eth0
# set GID/Group Name
SNORT_GID=snort
# other options
OPTIONS="-D"
# End of configuration
```

Then change directory to /etc/init.d and type:
chmod 755 snort (the file you just created, or copied from the contrib folder and modified)
cd /etc/rc3.d
ln -s ../init.d/snort S99snort
ln -s ../init.d/snort K99snort
cd /etc/rc5.d
ln -s ../init.d/snort S99snort
ln -s ../init.d/snort K99snort

cd ../init.d
chmod +x snort

Setting up the database in MySQL

Type /usr/local/mysql/bin/mysql (for deafault path of mysql specific bioneries) then press enter. And then do the following

mysql> SET PASSWORD FOR
snort@localhost=PASSWORD('new_password');

Explanation: [The above step sets password 'new_password" for user "snort" which has permission for the Snort's database on the host "localhost"]

>Query OK, 0 rows affected (0.16 sec)
mysql> create database snort;
>Query OK, 1 row affected (0.01 sec)
mysql> grant INSERT,SELECT on root.* to snort@localhost;
>Query OK, 0 rows affected (0.01 sec)
mysql> SET PASSWORD FOR
snort@localhost=PASSWORD('new_password');
>Query OK, 0 rows affected (0.12 sec)
mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to
snort@localhost;
>Query OK, 0 rows affected (0.01 sec)
mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to
snort;
>Query OK, 0 rows affected (0.01 sec)
mysql> exit
>Bye

From the Snort 2.2.0 source directory, execute the following command
/usr/local/mysql/bin/mysql -u root -p < ./contrib/create_mysql snort
Enter password:
(Enter the "new_password" which you have given earlier.)
Then install the extra DB tables using the following command from the contrib directory (you will need to cd to contrib)
zcat snortdb-extra.gz |/usr/local/mysql/bin/mysql -p snort
Enter password:
(Enter the "new_password" which you had given earlier.)
mysql> SHOW DATABASES;

(You should see the following)
+------------+
| Database
+------------+
| mysql
| snort
| test
+------------+

```
3 rows in set (0.00 sec)
mysql> use snort
>Database changed
mysql> SHOW TABLES;
+-----------------+
| Tables_in_snort
+-----------------+
| data
| detail
| encoding
| event
| flags
| icmphdr
| iphdr
| opt
| protocols
| reference
| reference_system
| schema
| sensor
| services
| sig_class
| sig_reference
| signature
| tcphdr
| udphdr
+-----------------+
19 rows in set (0.00 sec)>
exit
```

Snort and Reporting

With Snort running, it generates large number of events and alerts. These have to be analyzed and intrusion analysts need to be constantly referring and observing the events occurring from various sources. This directly demands software, which automates this process and presents reports, which can provide graphical representations about the events and summaries the same. One such reporting tool is ACID ie Analysis Console for Intrusion Detection. It is a PHP-based analysis engine to search and process a database of security events generated by various IDSes, firewalls, and network monitoring tools. It can be downloaded from Analysis Console for Intrusion Databases, http://acidlab.sourceforge.net/

Install JPGraph:

Change directory (cd) to where you have downloaded and stored the files.

```
cp jpgraph-1.16.tar.gz /usr/local/apache/htdocs
cd /usr/local/apache2/htdocs
tar –xvzf jpgraph-1.16.tar.gz
rm –rf jpgraph-1.16.tar.gz
```

Installing ADODB:
Change directory (cd) to where you have downloaded and stored the files.
cp adodb453.tgz /usr/local/apache2/htdocs/
cd /usr/local/apache2/htdocs
tar -xvzf adodb453.tgz
rm –rf adodb453.tgz

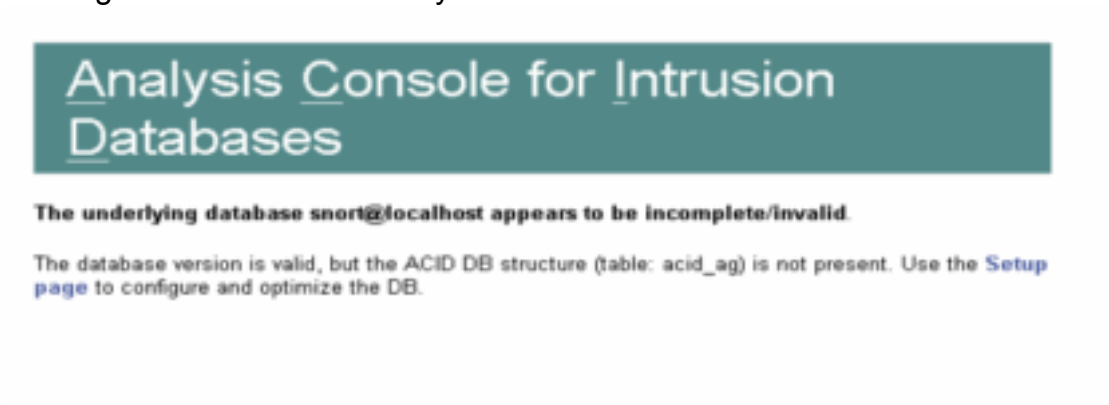
Installing and configuring ACID
Change directory (cd) to where you have downloaded and stored the files. cp acid-0.9.6b23.tar.gz /usr/local/apache2/htdocs
cd /usr/local/apache2/htdocs
tar –xvzf acid-0.9.6b23.tar.gz
rm -rf acid-0.9.6b23.tar.gz

Configuring ACID
Go to the /usr/local/apache2/htdocs/acid/ directory and edit the acid_conf.php file. It should look like this (except the password)

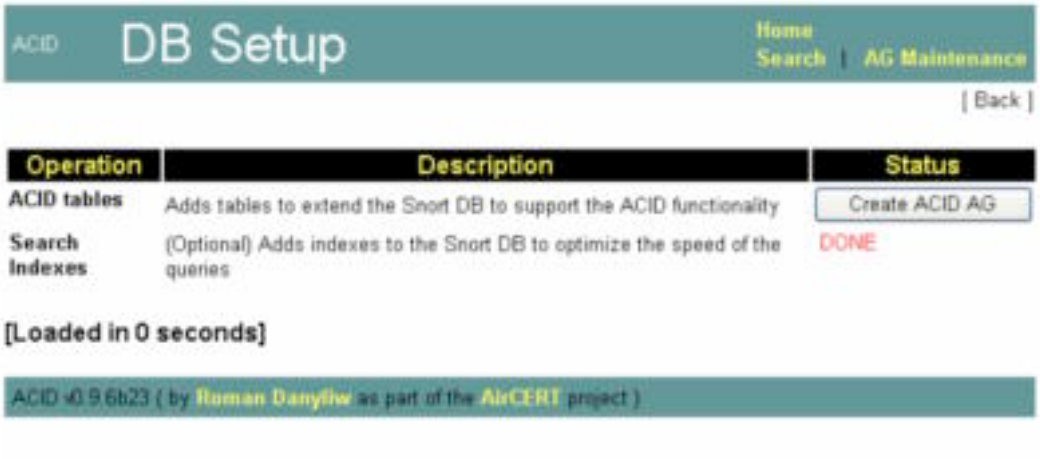
```
 $DBlib_path = "/usr/local/apache2/htdocs/adodb";
/* The type of underlying alert database
*
* MySQL : "mysql"
* PostgresSQL : "postgres"
* MS SQL Server : "mssql"
*/
$DBtype = "mysql";
/* Alert DB connection parameters
* - $alert_dbname : MySQL database name of Snort alert DB
* - $alert_host : host on which the DB is stored
* - $alert_port : port on which to access the DB
* - $alert_user : login to the database with this user
* - $alert_password : password of the DB user
* This information can be gleaned from the Snort database
* output plugin configuration.
*/
$alert_dbname = "snort";
$alert_host = "localhost";
$alert_port = "";
$alert_user = "snort";
$alert_password = "new_password";

/* Archive DB connection parameters */
$archive_dbname = "snort";
$archive_host = "localhost";
$archive_port = "";
$archive_user = "snort";
$archive_password = "new_password ";

$ChartLib_path = "/usr/local/apache2/htdocs/jpgraph-1.16/src";
/* File format of charts ('png', 'jpeg', 'gif') */
```

$chart_file_format = "png";

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



Click on the “Setup Page” hyperlink to create the tables that Acid uses, then you will see the following.



Then click “Create ACID AG” Button

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

Referred the above installation procedure from
Harper, Patrick,”Snort, Apache, SSL, PHP, MySQL and ACID install on Fedora Core 1 – From Source”
http://www.ntsug.org/docs/Snort_SSL_Acid_FC1_From_Source.pdf

Snort Usage

The following section discusses Snort and its usage in different modes and also how new Snort's rules should be written and the care taken in writing these rules.

How to Run Snort in Different Modes

Snort has many command line options that are very useful for starting the Snort in different modes. You can use **snort -?** to display the command line options. Some commonly used command line options and running snort in different modes are listed below

- -a this option sets snort in alert mode
- -v this option shows TCP/IP packet headers.
- -d shows the application data
- -e shows data link layer headers
- -l sets the directory where snort logs alert.
- -D to run snort in the background
- -c to specify the location of the snort configuration file(snort.conf)

Sniffer Mode

To see the Packet Headers only use **./snort –v**
To see application data and packet headers **./snort -vd**
To see data link layer headers, data and packet headers
./snort -vde

Packet Logger Mode

To Log all the packets in the specified destination dir (dest_dir) in a hierarchy based upon the host ip address use
./snort -dev – l <dest_dir>

To Log all the packets in the specified destination dir (dest_dir) in binary mode **./snort -l <out_dir> -b**

To log the packets related to a subnet
./snort -dev -l <dest_dir> -h <subnet>
(E.g. ./snort -dev -l ./log -h 192.168.1.0/24)

NIDS Mode

To run snort in NIDS Mode specify the Snort configuration file (Snort.conf by default it is in /etc/snort directory) with the command line option **-c**
E.g. snort -dev -l <dest_dir> -c /etc/snort/snort.conf
The above Intrusion Detection with Snort addition to acting as NIDS.
Reference: Snort man page
Snort Users Manual
http://www.snort.org/docs/snort_manual/node1.html

Snort Rule writing

Intrusion Analysts should practice writing effective Snort rules
An important paper “Intrusion Detection Systems with Snort Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID” by Rafeeq Ur Rehman,at
http://www.informit.com/content/downloads/perens/0131407333.pdf
describes these essential aspects of writing Snort rules

- A message part using the msg keyword.
- Rule classification, using the classification keyword.
- Use a number to identify a rule with the help of the sid keyword.
- If the vulnerability is known, always use a reference to a URL where more information can be found using the reference keyword.
- Always use the rev keyword in rules to keep a record of different rule versions.

Snort Testing

Functionality testing of snort is required to ensure the proper functioning of the features what snort claims to have. Test cases can be formed to cross check the functionalities and tools will help to realize these test cases. There have been Security laboratories like "The NSS Group" (http://www.nss.co.uk) established only to test Security products like IDS, firewall etc. The goal of such labs is to evaluate the IDS and standardize the testing methodology. Their are lot of public domain as well as commercial tools available to carry out and realize these test cases.

The hacker community mostly makes use of the same publicly available tools to its full strength to exploit and intrude into the Network and System. The following section discuses some of the tools and the task it performs
Nmap : Nmap is Network Exploration tool and security scanner which reveals open, filtered or closed ports. It also has the ability to identify Remote OS by TCP/IP Fingerprinting. Nmap does network scans like Vanilla TCP connect() scanning, TCP SYN (half open) scanning, TCP FIN, Xmas, or NULL (stealth) scanning, UDP raw ICMP port unreachable scanning, etc (Give reference Nmap man) to scan the network.

Nessus : Nessus is a open source vulnerability scanner, which provides the details about the vulnerability existing in the remote hosts. The report generated can determine whether any hacker may break into or misuse the vulnerabilities identified.

Nemesis: Nemesis is a command line, Packet Builder tool. It can be useful for easy injection of packet streams using simple shell scripts. It supports protocols like ([R]ARP, DNS, ICMP, IGMP, OSPF, RIP, TCP, UDP), and packets can be injected on either Layer 2 or Layer 3
Spak : Spak (Send PAcKet) is a collection of tools that can be used to generate and/or send arbitrary packets to a socket. Module generates TCP/IP packets by a shell pipe (IP, TCP, UDP).

whisker: It is Scriptable language that is tailored to do lots of flexible web scanning. It is very stealthy and Includes over 200 checks. It includes many anti-IDS tactics, brute force user names, brute force basic authentication guessing, now uses perl modules if available for extra speed, HTTP return values can be redefined, can now be used as a CGI, html output, SSL support and more vulnerabilities in the scan.

Apart from the above mention tools, there are lot of other tools which help the hacker in learning about the network, exploiting the system vulnerabilities and intruding into the network. Some of the widely known tools are icmpquery, sara, strobe, cheops-ng, SpiderMap, pof, ftpcheck, ftpscan, RvScan, SAINT, Nitko, cgi-check99, Stealth, Snot, Stick etc. It is very important to know the features and functioning of each of these tools to do thorough testing of the snort and to make the test cases realize.

By using the above tools for testing the snort IDS, one can cross check some of the functionalities of snort like PortScan Detection, Stream reassembly, Fragmented attacks etc. Using these tools one can try to adopt some of the evasion techniques and ensure that the snort is capable enough to detect those attack.
Reference
http://www.isecom.org/projects/toolsandtemplates.shtml

Snort & Some Useful Plugins

Front Ends - Open Source

- ACID - ACID is a php web-based front end to the mysql database http://acidlab.sourceforge.net/
- Base – BASE is the Basic Analysis and Security Engine. It is based on the code from the Analysis Console for Intrusion Databases (ACID) project. This application provides a web front-end to query and analyze the alerts coming from a SNORT IDS system. http://base.secureideas.net/
- SNORTER - SNORTER tool for the network intrusion detection system is an HTML reporting http://shweps.free.fr/snorter.html
- Sguil - sguil is a tcl based front end that uses barnyard http://sguil.sourceforge.net/
- Openaanval - a php, web browser based front end to the mysql database http://www.aanval.com/

Other Plugins

Barnyard

Barnyard is a output spool reader for Snort. This program decouples output overhead from the Snort network intrusion detection system and allows Snort to run at full speed.
http://sourceforge.net/projects/barnyard

Oinkmaster

Oinkmaster is simple but useful Perl script released under the BSD license to help you update/manage your Snort rules and disable/enable/modify certain rules after each update http://oinkmaster.sourceforge.net/about.shtml

Securing Snort, the NIDS Itself

It is very important to protect the integrity of the systems responsible for monitoring and maintaining the security of the network. A compromised NIDS sensor is a big threat to the security of the Organization since an IDS sensor would have access to the entire network traffic (including the to and from traffic of the critical servers).

There are three major security issues related to critical systems:

CERT defines three tenets of security as

Confidentiality—Maintaining the confidentiality of information stored system . This includes

- ensuring that only authorized users can access the services and information
- ensuring that authorized users can access only the services for which they are authorized

Integrity—Maintaining the integrity of information stored on the critical systems. This includes ensuring that you can recognize and recover from breaches of integrity.

Availability—Maintaining the availability of the services. This includes

- ensuring that services are uninterrupted even when there are hardware or software failures or during routine system maintenance
- ensuring that you can recognize and recover from security incidents n a timely manner

Steps to Secure the Snort NIDS machine

OS Security

- Organizations can use Secure Patch Management softwares to manage the patch process
- Install and use Bastille scripts according to the OS and platform
  Refer Bastille Linux http://www.bastille-linux.org for more details

Secure The Sensor Physically

The host systems on which the sensor and it's associated services are running should be physically secure. Make sure that only authorized people will have physical access to the place where these systems are situated. Implement the best possible (which is affordable to your organization) access control mechanism to restrict the access of the unauthorized people.

Enforce a Strong Authentication mechanism

Use stronger authentication mechanisms. Force the users to select long and complex passwords and make sure that the selected passwords changed periodically. Try to crack the passwords of the users with different password cracking tools to search for weak passwords and force the users to change the passwords. Configure the authentication mechanism of the system to lock the system after a certain number of failed attempts (maximum three or four attempts)

Enforce the principle of Least Privileges

Enforce the principle of the least privileges on the systems where Snort and its associated services are running. User accounts should not be created on this system except those that are absolutely necessary and the users should be given only the least privileges necessary to carry out his/her work.

Disable Unnecessary Services

Do not run any other services on this system, except the services, which are absolutely necessary to perform the business functions of the IDS. Make sure that services running on these machines are up to date and secure.

Apply Patches and Updates

New threats are discovered and patches are released by the vendors is a continuous process. Make sure that Operating system on which the Snort and its associated services are running patched with the latest release from the vendors.

SSH for Remote Access

Make SSH mandatory for the remote access as SSH uses an encrypted data transfer. [Protocols like Telnet, FTP, which sends clear text, should not be allowed].

Do not respond to ping

Configure the system (host systems where Snort is running) not to respond to the ping (ICMP echo type) Packet. This could actually hinder some network troubleshooting activities, and hence could be a tradeoff.

Use netfilter / iptables (in Linux)

If the Snort is running on a Linux machine, use ipfilter/iptables to block the unwanted data. Snort will still be able to see all of the data.

Snort running in stealth Mode

Run the Snort in stealth mode. In Stealth mode the system only listens to the incoming traffic but does not send any packet out.

Run Snort on an interface with no IP Address

On Linux machines "ifconfig eth0 up" will bring up the interface without assigning the IP Address. The advantage is that, when a host does not have

an IP Address, nobody can access it. The user can configure another IP on eth1 using "ifconfig eth1 up" that can be used to access the Snort.

Securing Snort communication
Stunnel or Secure Tunnel is an open source package available from http://www.stunnel.org that provides you a secure tunnel between two hosts.

Get the latest version from the web site http://www.stunnel.org/download/
And install it on both the Snort machine and the database server. You have to run it on both the Snort machine (client) and the database server to establish a tunnel. On the database server, use the following command:

stunnel -P/tmp/ -p stunnel.pem -d <snort port> (port where snort is running) -r localhost:<database port>(port where database is running)

If the stunnel directory is not present in the PATH variable, use the full path name with the command.
On the Snort machine, use the following command:

stunnel -P/tmp/ -c -d <database port> –r <Server Where Database is running> : <snort port>
securing snort communication is referred from
Rehman, refeeq, "Intrusion Detection with SNORT: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID"
http://www.informit.com/content/downloads/perens/0131407333.pdf

Network Security Toolkit (NST) available fromhttp://nst.sourceforge.net/nst/ helps the security administrator with a comprehensive set of Open Source Network Security Tools to enhance the security. . This bootable ISO CD is based on Fedora core 2[19].

Snort kit
Users can refer the following websites, guides, and mailing lists and keep it in their lists of important references

Snort Website
Snort home Page http://www.snort.org/
Snort Manual
Snort Manual is Available At : http://www.snort.org/docs/snort_manual/

Snort Mailing Lists
Some Active Snort Mailing List References:
- Snort Mailing List http://www.snort.org/lists.html
- Secure point Snort Mailing List http://msgs.securepoint.com/snort/

[19]Linux Fedora website http://fedora.redhat.com/

- Neohapsis Snort Archive http://archives.neohapsis.com/archives/snort/
- http://marc.theaimsgroup.com/?l=snort-users

Tuning Snort

This section discusses on the measures that can help to run Snort effectively

How to improve the performance of Snort?

- Run Snort on high-end machines preferably with dual processors and good primary memory (512MB or above).
- Use Multiple Sensors for monitoring in a large network.
- Logically segregate the network into segments and one sensor can be used monitor a particular segment.

- Fine-tune the rule set. The snort administrator can enable/disable snort rules according to the network traffic.
  For example if the organization has blocked ftp (external to internal network) in the firewall the Snort rule set for *ftp.rules* can be disabled for network traffic coming from outside the firewall to inside the network

PART III SNORT: A NETWORK IDS FROM A RESEARCHER'S PERSPECTIVE

Evolving Trends in the field of Intrusion Detection

As the threat posed by the cyber-attacks is rapidly increasing, the need for an intrusion detection system is also increasing. Large number of IDS products has come into the market and there is no standard for evaluating Intrusion Detection technologies. Intrusion Detection technology is still in its infancy stage. The primary challenges to the intrusion detection are high rate of false alarms, inability to work in diverse environments, to detect attack in early stages and the inability to respond. Researches are going on to bridge these gaps. Intrusion Detection Systems themselves are prone to attacks. So, the latest IDS products must also be capable of defending themselves. The following are some of the evolving trends in the intrusion detection world.

Making Compatible with IPV6

Research works are going about the challenges posed by the introduction of IPv6 to the Intrusion Detection Systems and how to alter the present Intrusion Detection Systems to make compatible with the IPv6 protocol. A good reference paper on this topic is available on this topic at: Arrigo Triulzi, Intrusion Detection Systems and Ipv6
http://www.alchemistowl.org/arrigo/Papers/SPI2003-IDS-and-IPv6.pdf

Enterprise IDS

One of the main issues faced by most of the today's IDS is scalability. Monolithic or distributed IDS that collect the audit data and transmit it to a central host for processing are incapable of operating in a large enterprise network, with a vast number of hosts.

The solution to this problem involves the construction of the IDS through a layered architecture. In this layered architecture, every layer can perform analysis, summarisation and aggregation of data. These functionalities could be centralised or distributed, ie it can occur at various node in the layer. Every node in that model can operate by aggregating the audit data it receives from the lower layers and passing a summarized form to the upper layer. Thus, the actual detection of an intrusion can occur on any layer, with the simpler ones occurring at a lower layer and the advanced ones at a higher layer. A good reference paper on this topic is available at : Astithas Panagiotis, Koutepas Georgios, Moralis Athanassios, Maglaris Basil "SIDS - A system for enterprise-wide Intrusion Detection"
http://www.netmode.ntua.gr/~gkoutep/docs/sids-ISSEA.pdf

Adaptive IDS

A vast amount of research is ongoing in adaptive IDS using advanced concepts in Artificial Intelligence (AI) and data mining. This will enable the intrusion detection systems to learn new attack patters by continuously analyzing the network traffic and deducing the behavior whereby it becomes adaptive. A good reference paper is available at Lee, Wenke "Adaptive Intrusion Detection: a Data Mining Approach"

http://citeseer.ist.psu.edu/cache/papers/cs/22535/http:zSzzSzwww.csc.ncsu.eduzSzfacultyzSzleezSzpaperszSzai_review.pdf/lee00adaptive.pdf

Immune IDS

Immunologists have traditionally described the problem faced by the immune system as the problem of distinguishing "self" from the dangerous "other"(or "nonself") and eliminating the dangerous nonself. The problem of protecting the computer systems from malicious intrusions can similarly be viewed as the problem of distinguishing self from nonself. A good reference paper on computer immune system is available at *Somayaji, Anil, Hofmeyr,* Steven & Forrest, Stephanie "Principles of computer immune system" http://citeseer.ist.psu.edu/cache/papers/cs/877/http:zSzzSzwww.cs.unm.eduzSz~immseczSzpublicationszSzprinciples.pdf/principles-of-a-computer.pdf

Some good reference papers about the immune IDS are available at Security Research Group (SRG) Home Page.
http://www.cs.plu.edu/pub/faculty/spillman/SRG/ids.htm

Intrusion Prevention System (IPS)

IPS is a device (hardware or software) that has the ability to detect attacks, both known and unknown, and prevents the attack from being successful. Intrusion Prevention systems are proactive defence mechanisms designed to detect malicious packets within normal network traffic (something that the current breed of firewalls do not actually do, for example) and stop intrusions dead, blocking the offending traffic automatically before it does any damage rather than simply raising an alert as, or after, the malicious payload has been delivered.
Similar to an IDS classification, there are two Types Of IPS

HIPS

Host IPS relies on agents installed directly on the system being protected. It binds closely with the operating system kernel and services, monitoring and intercepting system calls to the kernel or APIs in order to prevent attacks as well as log them.

Network IPS (NIPS)

The Network IPS combines features of a standard IDS, an IPS and a firewall, and is sometimes known as an *In-line IDS* or *Gateway IDS (GIDS)*.NIPS has at least two network interfaces, one designated as *internal* and one as *external*. As packets appear at the either interface they are passed to the detection engine, at which point the IPS device functions much as any IDS would in determining whether or not the packet being examined poses a threat.

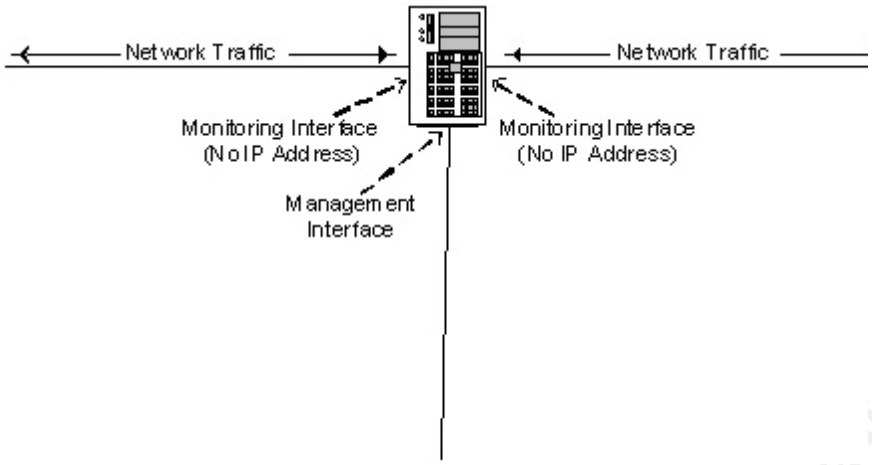


Figure Network Intrusion Prevention System with 2 NICS taken from Desai,Niel "Intrusion Prevention Systems: The next step in the evolution of IPS" Feb 27, 2003 http://www.securityfocus.com/infocus/1670

However, if it should detect a malicious packet, in addition to raising an alert, it will discard the packet and mark that flow as bad. As the remaining packets that make up that particular TCP session arrive at the IPS device, they are discarded immediately. Legitimate packets are passed through to the second interface and on to their intended destination.

Minimum Requirements of Intrusion Prevention Systems

In-line Operation

Only by operating in-line can an IPS device perform true protection, discarding all suspect packets immediately and blocking the remainder of that flow

Reliability and Availability

Should an in-line device fail, it has the potential to close a vital network path and thus, once again, cause a DOS condition. An extremely low failure rate is thus very important in.

Low latency and High performance

When a device is placed in-line, it is essential that its impact on overall network performance is minimal. Packet processing rates must be at the rated speed of the device under real-life traffic conditions, and the device must meet the stated performance with all signatures enabled. This is very important for the IPS working in the gigabit networks.

Detection Accuracy

It is imperative that the quality of the signatures is beyond question, since false positives can lead to a Denial of Service condition

Advanced alert handling and forensic analysis capabilities

Once the alerts have been raised at the sensor and passed to a central console, someone has to examine them; correlate them where necessary, investigate them, and eventually decide on an action

References

NSS Group Website "Intrusion prevention systems (IPS) " jan 2004 http://www.nss.co.uk/WhitePapers/intrusion_prevention_systems.htm

Desai, Niel "Intrusion Prevention Systems: The next step in the evolution of IPS" Feb 27, 2003 http://www.securityfocus.com/infocus/1670

IDS & IPS Scenario: Can Snort, IDS is still useful?

Both IDSs and IPSs have their own places. Since IPS blocks the traffic and tries to prevent attack, there is a risk involved in blocking legitimate traffic assuming it to be intrusive. IPS can block only those intrusions that can be detected with perfect surety. But, IDS can alert an intrusion of low severity based on which some response actions can be taken. SNORT 2.3.0RC1 claims to have added IPS functionality. SNORT. Also, in some cases, it is better that response is taken by a human (system administrator) rather than by an automated system.

Standards and Consortium

Standards are essential in the area of security to promote interoperability between the various solutions and vendor products. To take up the task, Internet Engineering Task Force (IETF) has formed a working group called the Intrusion Detection Working Group (IDWG). IDWG has published its specifications for a standard alert format (IDMEF) and a standard transport protocol (IDXP). Currently there are drafts available for IDXP protocol, which is soon to evolve as a RFC. (Request For Comments).

Common Vulnerabilities and Exposures (CVE) http://www.cve.mitre.org/ is working towards the standardization of signatures, which is gaining universal acceptance amongst security solution providers.

The leading commercial IDS vendors have joined together to form the ICSA Intrusion Detection Systems Consortium (IDSC), towards improving system and network security.

REFERENCES

[1] Snort website http://www.snort.org/

[2] Snort FAQ http://www.snort.org/docs/FAQ.txt

[3] Chuvakin, Anton, " Complete Snort-based IDS Archetecture", Nov 6, 2002 http://www.securityfocus.com/infocus/1640

[4] Wreski, Dave & Pallack, Christopher "Network Intrusion Detection Using Snort" 6/19/2000 http://www.linuxsecurity.com/feature_stories/feature_story-49.html

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

[6] Rehman, refeeq, "Intrusion Detection with SNORT: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID" http://www.informit.com/content/downloads/perens/0131407333.pdf

[7] Harper, Patrick," Snort, Apache, SSL, PHP, MySQL and ACID install on Fedora Core 1 –FromSource "http://www.ntsug.org/docs/Snort_SSL_Acid_FC1_From_Source.pdf

[8] Base, Rebeca and Mell, Peter "Intrusion Detection Systems" http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf

[9] Snort User Manual http://www.snort.org/docs/snort_manual/

[10] LibPcap Packet capture tool. Available at http://www.tcpdump.org/

[11] WinPcap Packet capture tool. Available at http://windump.polito.it/

[12] . Berkeley Packet Filter Packet capture library available at http://www-nrg.ee.lbl.gov/

[13] RFC 1180 - TCP/IP tutorial http://www.faqs.org/rfcs/rfc1180.html

[14] RFC 1858 –Security Considerations for IP Fragment Filtering. http://www.faqs.org/rfcs/rfc1858.html

[15] Hacker tool for creating malicious fragmented packets http://www.monkey.org/~dugsong/fragroute/

[16] Hypertext Transfer Protocol RFC http://www.faqs.org/rfcs/rfc1945.html

[17] Ollamann, Gunter "URL Encoded Attacks using the common web browser" Technical Info. http://www.technicalinfo.net/papers/URLEmbeddedAttacks.html

[18] Transmission control protocol(TCP) RFC http://www.faqs.org/rfcs/rfc793.html

[19] User Datagram Protocol (UDP)RFC http://www.faqs.org/rfcs/rfc768.html

[20] Telnet protocol Specification (RFC) http://www.faqs.org/rfcs/rfc854.html

[21] File transfer protocol (RFC) http://www.faqs.org/rfcs/rfc959.html

Snort Detection Engine And Pattern Matching Algorithm

[22] Norton Marc, " Optimizing pattern matching for intrusion Detection" September 2004 http://www.sourcefire.com/products/downloads/secured/sf_OPMforID.pdf

[23] Roelker, Dan, "HTTP IDS Evasions Revisited ", April 2004http://www.sourcefire.com/products/downloads/secured/sf_snort20_detection_rvstd.pdf

[24] Norton Marc, Roelker Daniel " Snort 2.0 Rule Optimizer" April 2004 http://www.sourcefire.com/products/downloads/secured/sf_snort20_ruleop.pdf

[25] Norton, Mark & Roelker, Dan " Snort 2.0 - Multi-Rule Inspection Engine ", April2004 http://www.sourcefire.com/products/downloads/secured/sf_snort20_HPMRIE.pdf

[26] Fist mike, Varghese george "Fast Content-Based Packet Handling for Intrusion Detection" May 2001 http://public.lanl.gov/mfisk/papers/ucsd-tr-cs2001-0670.pdf

[27] Whitehats website http://www.whitehats.com/ids/

[28] Common Vulnerabilities & Exposures, CVE, http://www.cve.mitre.org/]

Linux Security

[29] Bastille Linux http://www.bastille-linux.org

[30] Rushmore George, "Tips on basic Linux server security" http://www.net-security.org/article.php?id=109

[31] Stancin Aleksandar , "Securing Linux"http://www.net-security.org/article.php?id=111

[32] NSS Group Website "Intrusion prevention systems (IPS) " jan 2004 http://www.nss.co.uk/WhitePapers/intrusion_prevention_systems.htm

[33] Desai,Niel "Intrusion Prevention Systems: The next step in the evolution of IPS" Feb 27, 2003 http://www.securityfocus.com/infocus/1670

[34] Common Vulnerabilities and Exposures (CVE) http://www.cve.mitre.org/

[35] ICSA labs http://www.icsalabs.com/html/communities/ids/index.shtml

[36] Intrusion Detection Assessment http://www.icsalabs.com/html/communities/ids/whitepaper/Intrusion1.pdf

[37] Experiences Benchmarking Intrusion Detection Systems, Marcus Ranum http://www.nfr.com/resource/downloads/ExperiencesBenchmark

ingIDS.pdf

[38] Mike Barket, Intrusion Prevention Systems http://www.nfr.com/resource/downloads/SentivistIPS-WP.pdf

[39] Strengthen Control System Security By Preventing Network Attacks, NFR Security
NFR Security offers strategies for upgrading security to DCS / SCADA systems
and protecting critical infrastructure
http://www.nfr.com/resource/downloads/SCADA-Control_System_Guide.pdf

[40] Greg Taleck Ambiguity Resolution via Passive OS Fingerprinting http://www.nfr.com/resource/downloads/AmbiguityResolution.pdf

[41] Andre Yee, The Intelligent IDS http://www.nfr.com/resource/downloads/The_Intelligent_IDS.pdf

[42] Shankar, Umesh.: Active Mapping: Resisting NIDS Evasion Without Altering Traffic http://www.cs.berkeley.edu/~ushankar/research/active/activemap.pdf

[43] Beyond the Perimeter: Enterprise-wide Intrusion Prevention http://wp.bitpipe.com/resource/org_1046366622_812/IPS_White paper.pdf

[44] Linux RH Patches and vulnerabilities http://www.linuxsecurity.com/advisories/redhat.html

[45] Roesch Martin “Snort - Lightweight Intrusion Detection for Network by” http://www.snort.org/docs/lisapaper.txt

[46] http://www.isecom.org/projects/toolsandtemplates.shtml

[47] The NSS Group http://www.nss.co.uk

Securing Critical Systems

[48] Goldschmidt Robert “Securing Your Organization:A Technical Best Practices Overview” march 13 2002 http://www.ehcca.com/presentations/HIPAAWest2/goldschmidt.pdf

[49] Best Practices in Network Security http://www.networkcomputing.com/1105/1105f2.html

[50] SANS Security Policies http://www.sans.org/resources/policies/

[51] CERT Security Improvement module http://www.cert.org/security-improvement/

APPENDIX

Appendix A Difference Between HIDS and NIDS

HIDS	NIDS
Monitors Specific System Activity. This includes user and file access activities.	Monitors the Network activities.
Functions well in the network based encrypted environments (because by the time HIDS sees the incoming data it has been decrypted)	Because NIDS reads packet headers, (which a HIDS does not) It detects the attacks the HIDS do not. (Example for this attacks are different IP based Denial of Service Attacks (DOS)).
Detects only successful attacks. HIDS can detect attacks, which an NIDS can't (For example attacks from the keyboard of a critical machine does not pass through the network.	Detects unsuccessful attacks also (an NIDS placed outside of a Firewall can detect the attacks intended for resources behind Firewall.)
If the attacker is able to remove the logs of the successful attacks, HIDS might not be able to detect the attack (Since HIDS detects attacks using logs)	It is more difficult for an attacker to remove the logs in a NIDS.(Since it uses the live network traffic for real time attack detection)
HIDS is dependent on the host Operating System.	NIDS is independent on the host operating System

Appendix A.2 Configuration and Installation of MySQL, Apache, PHP

Installing PCRE:

Change directory (cd) to where you have downloaded and stored the files then execute the commands.

```
tar –xvzf pcre-5.0.tar.gz
cd pcre-5.0
./configure
make
make install
```

Install zlib:

Change directory (cd) to where you have downloaded and stored the files.

```
tar -xvzf zlib-1.2.2.tar.gz
cd zlib-1.2.2
./configure; make test
make install
```

Install Libpcap:

Change directory (cd) to where you have downloaded and stored the files.

```
tar –xvzf libpcap-0.8.3.tar.gz
cd libpcap-0.8.3
./configure
make
make install
```

Install MySQL

Create the user and group for MySQL with the following commands:

```
groupadd mysql
useradd -g mysql mysql –s /dev/null
```

In /root edit the .bash_profile file so the PATH line to read as follows:

```
PATH=$PATH:$HOME/bin:/usr/local/mysql/bin
```

Go to the directory you downloaded everything to, and use the following commands to install and configure MySQL.

Change directory (cd) to where you have downloaded and stored the files.

```
tar –xvzf mysql-4.0.20.tar.gz
cd mysql-4.0.20
./configure --prefix=/usr/local/mysql
make
make install
scripts/mysql_install_db
chown -R root /usr/local/mysql
chown -R mysql /usr/local/mysql/var
chgrp -R mysql /usr/local/mysql
cp support-files/my-medium.cnf /etc/my.cnf
```

Next, add the lines “/usr/local/mysql/lib/mysql” and “/usr/local/lib” to the /etc/ld.so.conf
file.

After you add the lines, run “ldconfig –v”, as root

Set MySQL to start automatically.
Copy the file "mysql.server" from the support-files subfolder (it is under the source for mysql. If you downloaded everything to /root/snortinstall, then the path will be /root/snortinstall/mysql-4.0.20/support-files) to the /etc/init.d folder and call it mysql (the command to copy it from the support-files directory is "cp mysql.server /etc/init.d/mysql")
Use the following commands to create symbolic links in the startup folders for run levels 3 and 5. MySQL will now start automatically when you boot up.

```
cd /etc/rc3.d
ln -s ../init.d/mysql S85mysql
ln -s ../init.d/mysql K85mysql
cd /etc/rc5.d
ln -s ../init.d/mysql S85mysql
ln -s ../init.d/mysql K85mysql
cd ../init.d
chmod 755 mysql
```

Installing and configuring Apache with PHP
Change directory (cd) to where you have downloaded and stored the files.

```
tar -xvzf httpd-2.0.52.tar.gz
cd httpd_2.0.52
./configure --prefix=/usr/local/apache2 --enable-so
make
make install
```

To start the web server use

```
/usr/local/apache2/bin/apachectl start
```

To stop the web server use

```
/usr/local/apache2/bin/apachectl stop
```

```
cd ..
tar -xvzf php-4.3.9.tar.gz
cd php-4.3.9
./configure --prefix=/usr/local/apache2/php –with
apxs2=/usr/local/apache2/bin/apxs --with-config-
filepath=/usr/local/apache2/php --enable-sockets --with-
mysql=/usr/local/mysql --with-zlib-dir=/usr/local --with-gd=[GD Graphics
Library install prefix]

make
make install
cp php.ini-dist /usr/local/apache2/php/php.ini
```

Apache 2.0.52 is now installed in the /usr/local/apache2 directory. Go into the /usr/local/apache2/bin directory and do the following commands:

```
cp apachectl /etc/init.d/httpd
cd /etc/rc3.d
ln -s ../init.d/httpd S85httpd
```

ln -s ../init.d/httpd K85httpd
cd /etc/rc5.d
ln -s ../init.d/httpd S85httpd
ln -s ../init.d/httpd K85httpd
(The above lines will add a start up script to the system for both run level 3 and 5)
To test the Apache – PHP install, create a file called test.php in the /usr/local/apache2/htdocs directory. Place the following line in the file "<?php phpinfo(); ?>" (without the quotes). Start Apache using **"/etc/rc5.d/S85httpd start**". Now use a web browser to look at the file (http://IP_Addess/test.php). It should give you info on your system, Apache, and PHP.

Referred the above installation procedure from
Harper, Patrick,"Snort, Apache, SSL, PHP, MySQL and ACID install on Fedora Core 1 – From Source"
http://www.ntsug.org/docs/Snort_SSL_Acid_FC1_From_Source.pdf