THE FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCE

A GRAPHICAL DISPLAY OF THE SNORT RULE SET

By

JOSEPH PETER BELANS

A Project submitted to the
Department of Computer Science
in partial fulfillment of the
requirements for the degree of
Masters of Science

Degree Awarded:
Spring Semester, 2003

MAJOR PROFESSOR
Lois Hawkes

COMMITTEE
Alec Yasinsac
Jeff Bauer

TABLE OF CONTENTS

ABSTRACT

Snort, a lightweight intrusion detection system, uses a pattern-matching scheme to alert on malicious traffic. Packet signatures are matched against predefined rules to determine if the traffic is valid. Like many other rule-based systems, a major disadvantage is that the rule sets can grow extremely large. To compensate for the lengthiness of the rule set, many graphical user interfaces (GUIs) have been designed to aid administrators in configuring Snort. However, a significant understanding of the rule base, the traffic to be monitored, and the Snort detection process is required to gain the benefits of such configuration tools. In order to get a better understanding of the detection process, users have requested an interface to graphically display the flow of the rules, without the underlying details. The remainder of this paper will give background on intrusion detection and describe a web-based interface tool that addresses the requests made by many Snort users, to dynamically show the flow of the Snort rule set.

INTRODUCTION

As we all know, computers are increasing in size and speed at an incredible rate. Keeping up with technology today is not an easy task. As technology grows, so do the problems that are associated with it. A by-product of technology has shown itself in the form of worms, Trojan horses, and many other intrusion techniques. These are issues that many systems administrators must deal with on a daily basis. Now more than ever, the need for security is on the rise.

Why do we need security? With the daily advancements in technology today, the rate at which intrusions take place is increasing tremendously. For example, in mid-2001, the Code Red worm was released and replicated approximately every 37 minutes. The Slammer worm, which was released late January 2003, had a doubling rate of 8.5 seconds. Slammer was reported to be the fastest growing worm in history to hit the Internet [6].

Attacks are changing on a daily basis. Once one attack is discovered and prevented, it may change ever so slightly to create another attack. Therefore, to list all the known attacks would take an entire document in itself. There are some types of common attacks that are seen on the network today. A few examples include protocol exploitations, worms, viruses, and denial of service (DoS) attacks. Protocol exploitations, which include simple port scans and IP spoofing, are one of the most common types of attacks and may be the basis for other attacks. As the use of web servers grows, so do the worms or viruses that attack the HTTP protocol. It seems that everyday there is a new Internet Information Server (IIS) virus appearing. DoS attacks directed at network connectivity are not very damaging but cause headaches for many

administrators. Syn flood attacks are an example of a frequently seen DoS attack. FTP and other known vulnerable applications are attacked daily on many systems across the Internet [4].

There are many types of tools available to alert users on, and/or protect users from, the dangers that lurk on an insecure network like firewalls, encryption, intrusion detectors, and intrusion protectors. Unlike firewalls and encryption that provide an overall layer of protection, intrusion detection and prevention systems provide information and/or precise protection from known and unknown attacks. Snort, a lightweight intrusion detection system, provides a mechanism to analyze known attacks. The Snort detection engine uses an extremely large rule set to pattern match against network traffic. For example, the latest set of rules provided with Snort includes 1,821 rules. As more vulnerabilities become recognized, the number rules provided will increase. With the rule set size increasing, so does the time and intervention required to understand what Snort is analyzing. It is the goal of this project to produce a tool that will help minimize the time it takes gain a full understanding of the flow of the Snort rule set. Therefore, this paper will describe the advantages and disadvantages of intrusion detection and prevention systems, an intrusion detection system titled Snort, and an extension to Snort to provide a better understanding to users of how Snort detects intrusions.

CHAPTER 1

NETWORK SECURITY

Everyday, alerts are presented to users concerning the release of a new worm or virus that has "hit" the Internet. Stories of web sites that have been defaced are now common dialog amongst administrators. The once safe and educational Internet has become one of the most insecure mediums available. As a result, network security is now one of the most regarded details applied to a local network. Associated with network security are *intrusion detection* and *intrusion prevention* systems.

1.1. Intrusion Detection Systems

Intrusion detection systems (IDS) provide detailed information about events that are occurring, or have occurred, on a given network. In general, intrusion detection is an analysis of the network traffic that may be entering or leaving the given network. The two detection methods used are *anomaly detection* and *misuse detection*. *Anomaly detection* models normal behavior. Any deviation of this model would be considered an attack. *Misuse detection* models abnormal behavior that would clearly indicate an attack. Some applications that currently use a form of intrusion detection engine are *honey nets* and *personal firewalls*. *Honey nets* are intentionally vulnerable systems created to attract users to use them in a malicious fashion. As users "hack and crack" at the honey net, information can be acquired to defend against the techniques used. *Personal firewalls* are a software-based firewall and IDS that can be applied to

smaller workstations to provide protection while accessing an insecure network. Software implementations began to appear when high-speed personal Internet access, like DSL and cable, was introduced. [5].

Intrusion detection systems have the advantage of not incurring any additional latency upon the network or its hosts. One way is to analyze the network traffic offline in the form a dump file. Another is to receive identical traffic via a hub or port mirroring on a switch. Real time monitoring of multiple systems can be achieved when a network interface card (NIC) is placed in promiscuous mode. In promiscuous mode, a NIC will listen to a wire and pick up any traffic that it hears, not just traffic that is destined to it. To improve security, a monitor can be isolated from the network by not assigning a routable address to the NIC. In essence, it can receive all traffic but will not send any traffic [5].

As the name states, intrusion detection systems "detect" traffic. Unfortunately, detection is all that is accomplished. There is no protection provided to the network or hosts that are being monitored. Also, as a result from continuous analysis, excessive log files may be created that will require some sort of manual intervention to analyze. Therefore, analysis of a network can become too time consuming to acquire a meaningful result [4].

1.2. Intrusion Prevention Systems

Intrusion prevention, on the other hand, does provide a means of protection for a given network and its hosts. The basis for the protection is built upon the same detection methods that were introduced in intrusion detection systems with some added complexity for speed and quality. Four approaches to intrusion prevention include *stateful signature detection*, various types of *anomaly detection*, a software based *heuristic approach*, and a *hybrid approach*.

Stateful signature detection uses pattern matching, which requires prior knowledge of an attack to follow state within a series of packets by looking at relevant portions of the traffic where the attack could be perpetrated. A software based *heuristic approach* is similar to an IDS using anomaly detection built upon a neural network. Combining various detection methods and signature detection creates a *hybrid approach* to gain the advantages of each type of detection method. An example of an intrusion prevention system is StormWatch which has a network interceptor built into its detection engine [3].

The obvious advantage to intrusion prevention systems over detection systems is that a layer of protection is introduced to the given network. However, the added protection may come at a price. If not configured correctly, the network may experience an added latency as a result of added processing to determine if an attack is eminent. Also, any false positives could greatly hinder a network by eliminating legitimate traffic that may resemble an attack. Therefore, as stated earlier, the added protection, if not implemented correctly, could have negative effects.

1.3. Components of an Intrusion Detection System

Although intrusion detection systems have evolved over the years, the same atomic building blocks can be found in every system. Each system will contain the following functionality: a *probe*, *monitor*, *resolver*, and *controller*. The *probe* is the basis for the entire system. Probes listen and decode network traffic into a more readable form. *Monitors* take the information provided by the probes, analyze it, and emit any necessary alerts. *Resolvers* take the necessary action based on the alert. *Controllers*, only apparent in distributed intrusion detection systems, act as a central point of administration.The original *probe*, when intrusion detection systems were log based, analyzed log files offline. In a networked environment, there are *host*

network and *promiscuous network-based* systems. *Host network-based* probes will decode traffic destined for an individual host. *Promiscuous network-based* probes will decode any traffic that it hears on a wire [5].

Monitors use anomaly detection and/or misuse detection to analyze the decoded network traffic. Anomaly and misuse detection behave like inverses of each other. Anomaly detection provides better intelligence in detecting variations of attacks but can produce an abundance of false positives. On the other hand, misuse detection minimizes false positives but lacks in the ability to provide information about new or morphed attacks. False positives and the ability to detect unknown attacks are two of the most noted problems in monitor design. Combining both methodologies can alleviate these problems and is seen in many systems [5].

Statistical modeling, a technique used in anomaly detection, uses some of the more well known mathematical processes like mean and standard deviation, the Multivarate model, and the Markov process model. An immune systems approach follows system calls within an application to create conditional models of normal behavior, error conditions, and attempted exploits. Protocol verification rigorously checks for malformed protocol fields. Violations of given protocols are key indications of an attack. File checking uses standard checksums to check the validity of the file. Taint checking assumes any input data provided is considered "tainted" until it is cleaned [5].

Misuse detection, in its most basic form, is achieved using expression matching. The expressions can be in a binary format, a character string within a web request, or any other type of pattern that may be found within a given packet. State transition analysis uses finite state machines to follow an attack. When a state machine reaches its final destination, it signifies an attack. Dedicated languages create specialized programs to find an attack. As events are read in,

the program will match the events against internal filters to determine if an attack has taken place [5].

Intrusion detection systems can take the form of four architectures: *monolithic*, *hierarchic*, *agent-based*, and *distributed*. *Monolithic* systems house all the basic components to monitor a single system. *Hierarchic* systems have a centralized controller to correlate data from multiple systems. *Agent-based* systems include multiple probes, controllers, and monitors. *Distributed* systems build on an agent-based architecture with added pattern matching to analyze multiple systems as a whole, not as individual systems.

CHAPTER 2

SECURITY TOOLS

Security tools aid in the design, implementation, monitoring, and testing of secure networks. Packet sniffers and port scanners are only a few of many tools available to administrators to test the security of a network. However, they are more than enough for a rogue user to compromise a network. In response, intrusion detection and prevention systems have been created. Tools like tcpdump, nmap, and Snort will be described below.

2.1. Tools of Protection or Mass Destruction

There are many other types of security tools. Unfortunately, most tools that are built to help an administrator can be used to create havoc. Port scanning is one of the most often seen attacks. Many attacks begin with a port scan to allow an attacker to determine what services are available on a given network or host. Nmap is a very popular port scanner that is free and comes pre-built on many variations of UNIX. Due to its availability, it is very widely used by many attackers. One of the many features that nmap provides is the ability to spoof the origin address. The functionality was added to allow administrators to create randomness in their port scans to find holes in their security schemes. As a result, attackers can now spoof themselves when using nmap in an attack. Another widely used tool is tcpdump, or some other form of packet sniffer. Packet sniffers were also built to provide a means for administrators to create a snap shot of the

traffic on the network. In the hands of an attacker, it is an easy way to find usernames and passwords to gain access in an insecure network.

As one can easily picture, security tools, in the wrong hands, can provide a significant amount of information to an attacker. Used correctly, one can provide a force field of security and understand completely the state of a host or network. Having the right information will give an administrator what he or she needs to defeat an attacker.

Another security tool that is available that can produce a very detailed picture of a host or network is Snort. Snort is a glorified version of tcpdump; a packet sniffer that has a detection engine to alert upon detection of abnormal traffic.

2.2. Snort Intrusion Detection System

2.2.1. Background

Snort is considered to be a cross-platform, lightweight intrusion detection system. A lightweight intrusion detection system can be deployed on any given system having minimal affects on the system. As per the GNU public license, Snort is free to use, change, and distribute. An open source library, libpcap, is the basis for the Snort technology. Snort compares relatively well to other commercial applications in that it is very inexpensive and trivial to implement. Many other applications may have an abundance of functionality compared to Snort, but, with complexity comes a price and a lot of additional work [2]. Some of the well-known companies that provide a variation of an intrusion detection and/or prevention system are Cisco [10] and Enterasys [11].

2.2.2. Primary Components

Snort, like any intrusion detection system, contains the three primary components: the probe, the detection engine, and the alerting and logging system. The probe, or packet decoder, makes up the first primary subsystem of Snort that is built upon the libpcap library. Libpcap interfaces with the NIC to decode network traffic. Without any additional changes or extensions, Snort can decode Ethernet, Serial Line Internet Protocol (SLIP), and raw Point-to-Point Protocol (PPP) packets. Ethernet traffic is typically seen in local area networks. SLIP and PPP are used to transport data over serial lines, for example, telephone lines between two modems. Depending on the logical location of Snort, the system can be configured as a host network or promiscuous network-based system. As traffic is read in from the network, the decoder will create pointers to packet data that is later analyzed by the detection engine. The detection engine uses a pattern matching approach to detect abnormal behavior and can then issue an alert. With the use of a third party plug-in, for example Spade, anomaly detection can be used to minimize the rule set by creating a picture of normal traffic and allowing the rule set to process further any packets that do not fit the picture [2].

Rounding out the overall architecture is the logging and alerting subsystem. Built into the Snort software are various logging and alerting mechanisms. Logs can be built in a binary or plain text format. Alerts can be sent to a text file, syslog, a samba pop-up window, or an SQL database [2]. Syslog is a logging mechanism that is available in most UNIX variants out of the box. A third-party application, Samba, provides various functionalities mainly that being the ability to mount a UNIX file system to a Microsoft Windows machine. One of the alerting mechanisms built into Samba is the ability to create a pop-up window on a user's machine for various informational purposes. Snort can use this functionality to alert an attack. Lastly,

various Simple Query Language (SQL) databases can be utilized to store alert data. Some of the many databases that Snort is pre-configured to use are MySQL, PostgreSQL, Oracle, and unixODBC-compliant databases [2].

Snort is very versatile in that it can take on many forms. Internally, Snort can act as a simple packet sniffer and display network traffic to standard out. Packet logging is also possible with Snort so that the information can be stored for later analysis. Once given a logging directory, Snort will decode the traffic and create a directory structure containing information pertaining to each host that is represented in the decoded traffic. Lastly, and in the most significant form, Snort can provide intrusion detection functionality [1]. As an intrusion detection system Snort can be configured as a monolithic, hierarchic, or agent-based system.

2.2.3. Snort Detection Engine

The heart of the Snort detection engine resides in the rule set that is provided by the user. As network traffic is read in, each packet will be pattern matched to the rules provided to determine if it matches a known attack. Each rule has a predefined format that allows for the most general to extremely complex rules. A general template that all rules will follow is:

<action> <protocol> <src IP> <src port> <direction> <dest IP> <dest port> <options>

Rules have two basic components: *rule headers* and *rule options*. *Rule headers* make up the first seven fields of the rule:

<action> tells the type of the rule

<protocol> is the type of traffic the rule matches

<src IP> and *<src port>* describe where the traffic came from

<direction> shows which direction the traffic is traveling that the rule will match

<*dest IP*> and <*dest port*> describe where the traffic is going

Each header contains common attributes found during a given attack. All the headers are found in the first dimension of the link list. *Rule options*, which are not required, provide the expressions on how the rule will match a packet. A list of options associated with each header compile the second dimension in the link list [1]. For example:

alert tcp any any -> 192.168.1.0/24 111
(content:"|00 01 86 a5|"; msg: "mountd access";)

The above rule will be used throughout the remainder of this section to provide an example for each field within the rule.

Rule headers are made up of five components: the *type* of rule, *protocol*, *source*, *destination*, and *direction*. There are five *types* of rules that make up the first field: *activate*, *dynamic*, *alert*, *pass*, or *log*. *Alert* rules make up the majority of most rule sets. It is how Snort notifies an administrator of an attack by way of the logging or alerting mechanism that is specified in the configuration. From the example, we see that the rule is an alert rule from the first field: ***alert*** *tcp any any -> 192.168.1.0/24 111(content:"|00 01 86 a5|"; msg: "mountd access";)*. *Log* rules log the packet to disk creating a directory structure describing hosts and the networks that the hosts reside on. *Pass* rules assume the traffic is legitimate and does nothing. *Activate* rules alert an administrator and activate another dynamic rule. *Dynamic* rules are dormant until activated and then become log rules [1].

The second field specifies the *protocol* that must be associated with a given packet to match the rule. By default, Snort's decoder will decode IP, TCP, UDP, and ICMP packets. Obviously, the example rule will match TCP packets: *alert* ***tcp*** *any any -> 192.168.1.0/24 111(content:"|00 01 86 a5|"; msg: "mountd access";)*. There are others that have been added with the use of pre-processors and other plug-ins [1].

The next two fields provide the *source* IP address and port number. The IP address field can contain a single IP address, a masked address, or the keyword "any" to specify any address will match. An IP address is specified using CIDR notation, a short-hand way of displaying the subnet mask associated with an address. For example, 192.168.1.0/24 translates into 192.168.1.0/255.255.255.0 which will include the addresses between 192.168.1.0 and 192.168.1.255. Negation is also allowed to specify any other address except the one(s) specified that will match the rule. Port numbers are specified in the same manor. Also, the same criteria will be used to specify the *destination* IP address and port number [1]. Any packet that is destined for an address within the 192.168.1 subnet and port 111 would match the example rule: *alert tcp **any any** -> **192.168.1.0/24 111**(content:"|00 01 86 a5|"; msg: "mountd access";)*.

The *directional* field will determine which direction the traffic must flow to match the rule. The majority of the rule set will use the source to destination operator (->) that will only match incoming traffic. This is the case with the example rule as it will only watch incoming traffic: *alert tcp any any -> 192.168.1.0/24 111(content:"|00 01 86 a5|"; msg: "mountd access";)*. With the use of the bi-directional operator (<>), one can follow a conversation between two machines [1].

Rule options provide the necessary information for the detection engine to process packets. There are an extremely large number of keywords that can be included in the rule options, and therefore, will not be completely described here. Two keywords that are most often used in rule options are *content* and *msg*. The *content* keyword provides an expression to match against the packet. Content can take the form of a binary string to a complex expression containing wild cards to match a text string contained within a packet. *Msg* describes the message that will be logged or sent to an administrator via the logging or alerting mechanism

specified in the configuration [1]. Combining these two keywords makes for a very simple rule but helps in our example: *alert tcp any any -> 192.168.1.0/24 111(content:"|00 01 86 a5|"; msg: "mountd access";)*.

2.2.4. Motivation for project

There are many tools available that extend or enhance the front and back end of Snort. For example, Webmin, a popular collection of miscellaneous tools, has a Snort front-end module that will assist an administrator in configuring the Snort rule set [9]. The Analysis Console for Intrusion Databases (ACID), converts data collected and stored in a database into a meaningful human readable form for analysis [8]. Unfortunately, there is nothing currently available to analyze the extremely large rule set, which is Snort's major draw back. As the number of rules increases from the most recent set of 1821 rules, so does the time it takes to understand the rule set. Once a rule is added to the rule set, depending on its location and expressions, it could possibly be matched against every packet that is intercepted by the probe. It becomes very apparent that without careful configuration of Snort, degradation of performance would be seen immediately. There may also be many other rules included in the set that are never matched. This requires unnecessary processing power that could be eliminated if one knew how the rule set was processed.

After browsing the mailing.unix.snort news group, it was apparent that a tool that would graphically show an administrator how Snort internally processes the rule set would be very useful. This determination was based on the many questions that were asked pertaining to the internal structure and functionality of the Snort detection engine. It seems there is a misunderstanding amongst many of the Snort user community about how the rule set is

processed by Snort. Understanding the detection engine, and the traffic flowing through it, would allow an administrator to configure Snort for better performance. Therefore, the remaining portion of this document describes such a user interface, one that will allow a user to create an image of how Snort will process the rules but without rule details. The interface provided gives the user the flexibility of viewing the entire rule set, a single rule, or a portion of the rule set. With the information provided by the tool, a user can make better decisions on how to configure Snort so that it more closely matches that type of traffic that is found on a given network. It can also provide the "visible" information that may help others that have a less technical background understand what is being detected on their network by seeing the rule set without the underlying details.

CHAPTER 3

SNORT DISPLAY TOOL

The following application, Snort Display, was created to provide a picture of how the Snort detection engine interprets the rule set to provide information as to how Snort processes the rule set. In order to correctly display how Snort works, two functions that are included in the Snort source code are used to dump the information as it is portrayed by Snort. With the use of the interface tool developed, a user may display all rules or a subset by narrowing down the criteria to display. Once a search criteria has been provided, the requested graph will be displayed. Information provided in the graph solves two problems associated with Snort.

Large number of rules with unclear interactions

One of the issues that many Snort users have been dealing with is that of how does Snort process the rule set. For example, the latest rule set includes 1821 rules that will require an amount of manual intervention to fully understand. It is well documented that Snort parses the rule set in a first come, first serve process. Where the mystery lies is in the intelligence that is built into Snort that combines similar rules to speed up processing. Unfortunately, the final tree structure may not match what is expected if one were to read the rule files in the order they are parsed. It is the goal of this project to graphically provide information that will speed up the learning process to allow a user to increase the performance of Snort.

High amount of processing required to pattern match network traffic

A well known problem concerning pattern matching IDS is that as the size of the rule set increases, so does the time to process the network traffic that is being monitored. Therefore, one way to minimize the processing time is to minimize the number of patterns that need to be matched. The information contained in the graph can be compared to the type of traffic which will be seen on a network and Snort can then be configured to increase the overall performance.

3.1. Overview

Snort Display

1) Parses the Snort rule set;
2) Dynamically creates the interface;
3) Creates the graph.

1) Parse the Snort rule set

In order to create the interface, Snort Display must first parse the Snort rule set and create a configuration file. Upon clicking the "Browse" button, the user can search for a Snort configuration file. A new browser window will open displaying a change rooted directory. One can then traverse through the displayed directory tree to find the file. Next, the user will click the "Load" button to load the Snort configuration file, parse the rule set, and create the Snort Display configuration file. The interface will then refresh with the first drop down box containing the default rules.

2) Dynamically creates the interface

Initially the five default rules are read into the first drop down box. If the user selects "All", the entire rule set will be displayed. By selecting a type of rule, the interface refreshes with the second drop down box with a list of recognized protocols. Again, by selecting "All", all rules that are of the selected type will be displayed. Otherwise, by selecting a protocol fills in the final drop down box with a list of the associated rules. From this point, there are three different views available:

1) The user can display all the associated rules by selecting "All";
2) A subset of the rules associated with the protocol may be display by supplying a number of rules to display along with the location of the selected rule;
3) A single rule associated with the protocol may be displayed.

3) Creates the path

The selection criteria supplied by the user will be passed to the display function. With the information provided by the user and the rule details provided by Snort, Snort Display will create the graph details. A point on the graph will be created for each rule type, protocol, rule, and option to be displayed. With the use of several gd library functions, the graph will be dynamically created. Finally, an HTML page containing the graph will be provided to the user.

3.2. Design

In general, the project involved designing an application that would create a graphical representation of the Snort rule set. The requirements that were set from the beginning were that it would be web based, interactive, and dynamic. A web-based application was decided upon for

portability and to allow for wide-spread use. Being that web servers are becoming an industry standard for supplying information, and that many of the third-party applications that interface with Snort are web-based, it seemed that a web interface would provide the best solution to this problem.

An interactive implementation was required so that different variations of the graph could be displayed based on the user's criteria.

Finally, to allow for the interactive capability, the application would have to dynamically create the graph display for various requests.

3.2.1. Initial Design

The initial design took on a basic implementation in order to create a prototype of the initial idea and create a basis for other suggested functionality. Some of the initial design components that are implemented in the final design are:

a "Browse" interface to search for a Snort configuration file;

the ability to "Load" the Snort configuration file;

three drop down boxes that will provide the required information to create the graph.

Once the initial design took shape, two of the main problems that were contemplated many times were:

how to provide the information in the graph;

which direction should the graph flow.

Initially, the information was to be portrayed via different icons representing each point on the graph. Unfortunately, with the number of rules that needed to be displayed, the graph became quite large to the point that the library could not create an image large enough. Therefore, an implementation that provided the same information in a smaller space was required.

The direction of the flow of the graph was questioned many times throughout the design processes. Initially, the graph flowed from left to right with the heads of each tree aligned on the left margin. This followed the traditional idea of scrolling up and down rather than left to right. Unfortunately, with the number of options associated with some of the rules, the graph grew from left to right very quickly. Therefore, the graph was rotated to align the head nodes across the top of the graph. Again, the display scrolled from left to right being the rule set is so large. It was determined that there was no clear cut solution as to which direction the graph should flow. As the number of rules increases, so does the graph in one direction, or, as the number of related options increases, so does the graph in the other direction. The question remains, which direction is going to grow the fastest? This is answered in the final implementation.

3.2.2. Final design

Prior to presenting the application for the first time, some of the flaws that were seen in the initial design were updated. In order to minimize the size of the graph, points, rather than icons, were created on the graph. Each point had relevant information that would be seen when moused over. As the number of points that were displayed decreased, the size of each node increased with more information being displayed graphically.

The other problem of the direction of the graph was also dealt with. It was assumed that as the number of rules increases, the number of related rules may stay the same. The idea is that the components of a network will remain the same but, the points of attack may change, therefore creating other rules that are similar to rules that already exist. Thus, the number of options will increase faster than the number of individual rules. So, to minimize the amount of left to right scrolling, the graph was created in a top-down fasion.

After presenting the initial design of the application with the additional changes, some of the comments and suggestions that were made were:

provide different colors and shapes to show a distinction between the types of nodes;

add a legend describing each color and shape;

provide a mechanism to display a portion of the rules, not just a single rule or all rules;

decrease the size of each point to minimize the size of the overall graph.

The colorization of the graph was very straight forward with the functionality provided by the graphics library. Unfortunately, creating a different shape for each node was not as trivial. When creating a larger point on the graph, the detail of each shape was obvious. However, once the point size was decreased, the shapes were less defined and were not clearly displayed. After adding the different colors to the graph a legend was added to the top of the graph. With the colorization and legend, the details of the graph were much easier to distinguish.

Finally, a suggestion was made to add the ability to display a portion of the graph. This functionality rounded out the application into its final version. By allowing the user to state the number of rules to display, and the location to place the selected rule, a portion of the graph can

be displayed once a user has narrowed the criteria to a specific type of rule and protocol. With the final revision at hand, it seemed that all the necessary requirements were fulfilled.

3.3. Snort Display components

3.3.1. The Interface

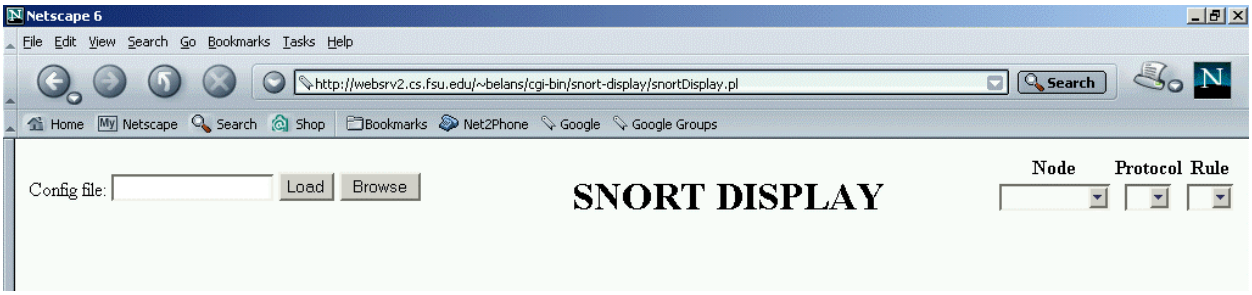

Figure 1 – The default interface

The interface provides access to the four main components of Snort Display: select a Snort configuration file via the "Browse" button, "Load" the Snort configuration file to create the Snort Display configuration file, generate a criteria for the graph via the three drop down boxes, and display the graph in the bottom half of the interface. Figure 1 shows an example of the top half of the interface that provides the interactive functionality available to the user. To find the Snort configuration file, an additional browser window will open when the user selects "Browse" (Figure 2) and display the contents of the change-rooted directory supplied when Snort Display was installed. The HTML code contained in the browse window is produced by another Perl script, browse.pl. As a user "clicks" on each directory, a new directory listing will be provided. Once the user selects a Snort configuration file, the full path will be entered into the interface.

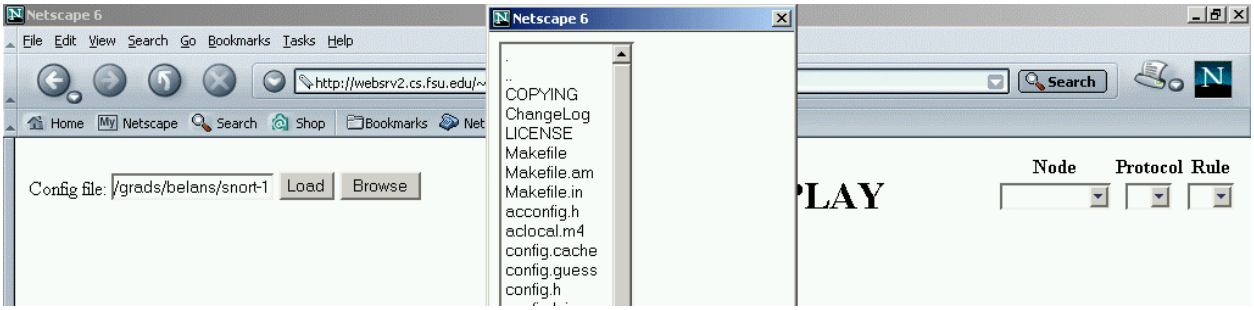

Figure 2 – Browse window

Upon selecting "Load", the Snort Display configuration file will be created based on how Snort parses the rule set. The first drop down box will be filled in with the five default rule types (Figure 3). As the user makes various selections, the interface will be re-displayed based on the new selection. A user will then have the ability to narrow down the search to a single type of rule, a single protocol, or a single rule by selecting one element from any or all of the three drop down select boxes. Or, the user can get the big picture by displaying the entire rule set all at once.

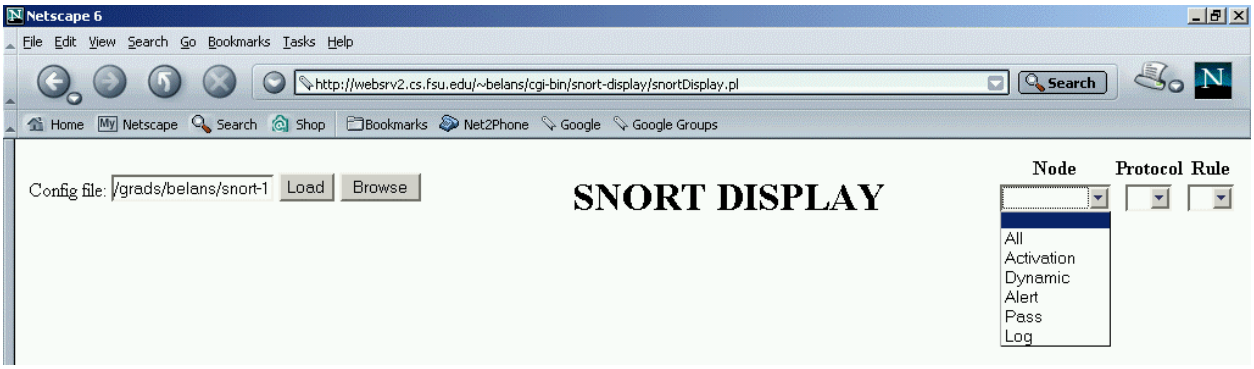

Figure 3 – Initialization of interface

3.3.2. The Display

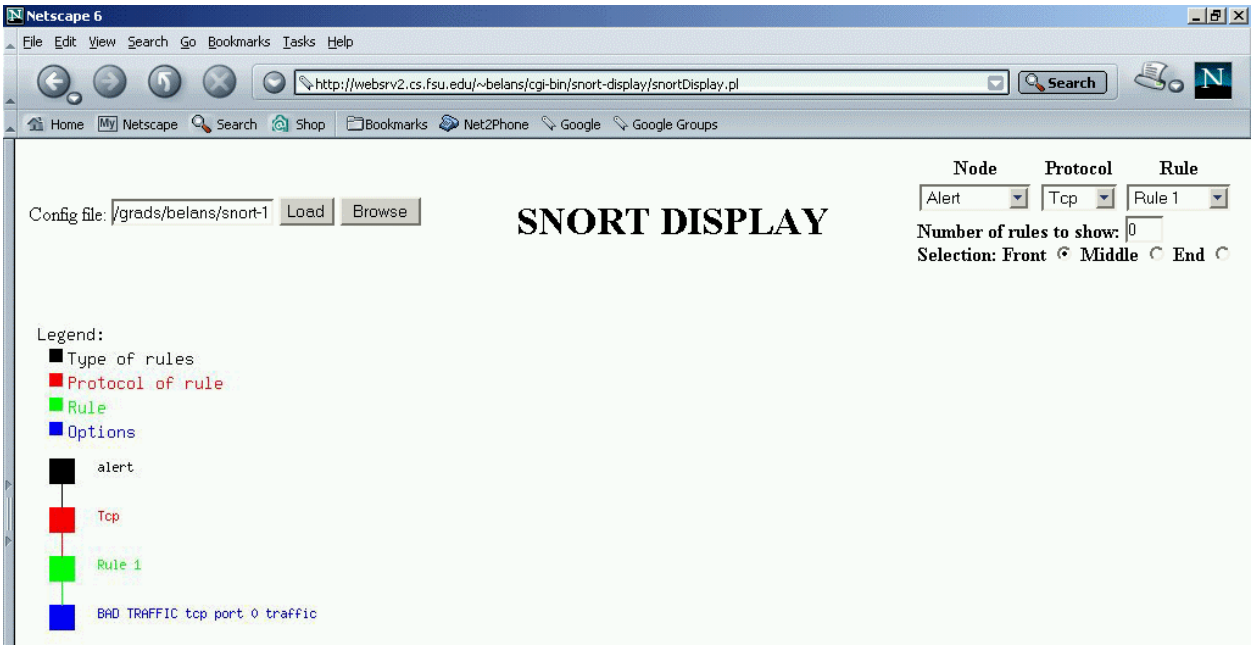

Figure 4 – Graph with additional data included

Based on the selection(s) made, the appropriate link lists will be searched and/or traversed to create and display the graph. Each point on the graph will represent a list, a protocol, a rule, or an option. Information associated with each point will be included in the dynamically created page. The information, by default, will be displayed as each point is moused over. As the number of points on the graph decreases, more information will appear on the graph itself (Figure 4).

Again, after demonstrating the initial design, many suggestions were made to improve functionality. The points, and the lines that are joining the points, were color coded to signify the type of point it represents. A legend is included at the top of the graph to signify the meaning of each color. Black represents the different types of rules. Red represents the protocol of the rule. Green represents the header information contained in each rule. Blue represents the options contained in each rule. Additional functionality was added to allow a user to determine

the number of rules that will appear in the graph. Once the user has selected a specific protocol, two other fields will appear (Figure 5):

a text box to enter the number of rules to display;

a set of radio buttons to determine the location of the selected rule, if any.

Based on the additional information from the user, and the added functionality, the user may create a very broad or narrow representation of the Snort rule set.

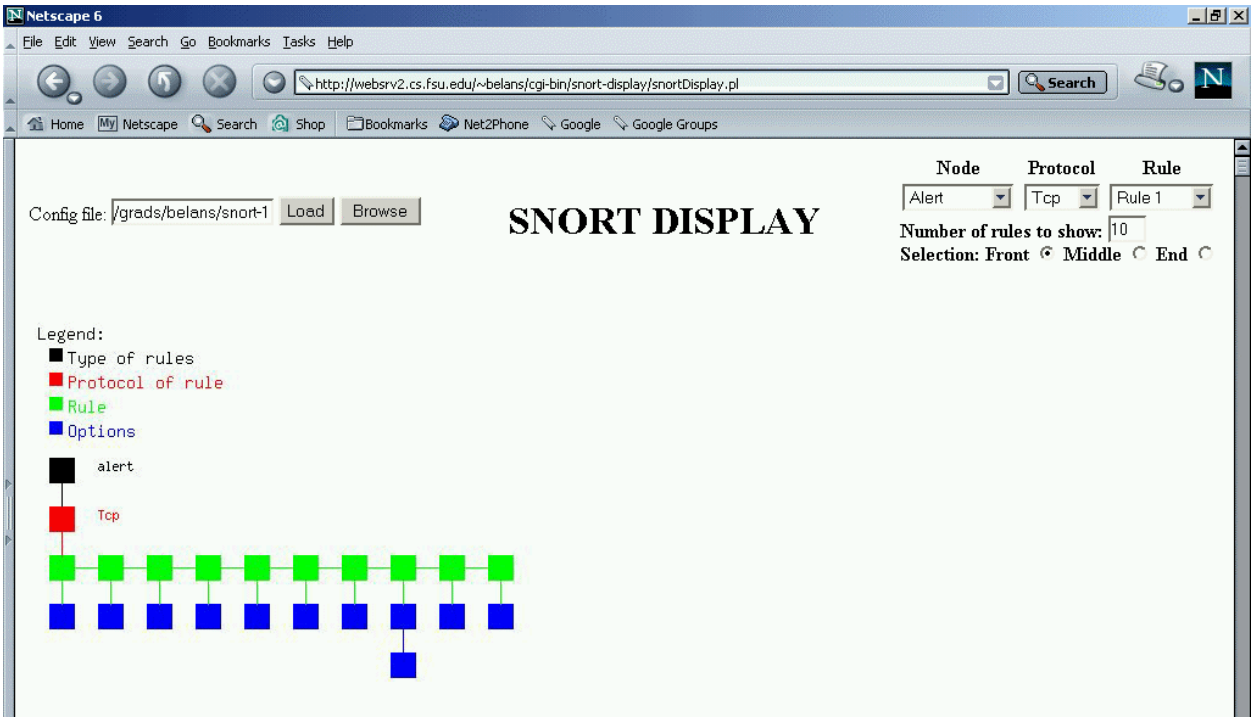

Figure 5 – Graph created based on selections

3.4. Snort Display implementation

3.4.1. Initial implementation

3.4.1.1. Snort Display functions

menu.pl()

Perl Script

Calls ***browse.pl()***, ***config.pl()***, ***display.pl()***, and itself

Creates the interface based on the Snort Display configuration file and user selections

Provides the basic mechanism for the user to interface with the application

browse.pl()

Perl script

Creates a browse window to find a Snort configuration file

config.pl()

Perl script

Calls ***config()***

Parses the form data to create the command line string to pass to config()

config()

C program

Calls ***InitNetmasks()***, ***InitPreprocessors()***, ***InitPlugIns()***, ***InitTag()***, ***InitVariables()***, ***CreateDefaultRules()***, ***ParseRulesFile()***, and ***CreateDataFile()***

Receives at least one argument, the snort configuration file

An optional output file may be provided, otherwise, the output will be sent to STDOUT

Snort dictated the language to use to create ***config()*** the Snort functions required are written in C

Function: ***CreateDataFile()***

Creates the Snort Display configuration file

display.pl()

Perl script

Calls ***display()***

Parses the form data to create the command line string to pass to ***display()***

display()

C program

Calls ***readFile()***, ***drawImage()***, ***drawNodes()***, ***drawLists()***, ***drawRules()***, ***drawOptions()***, ***freeImage()***

Receives the type of rule, a protocol, and a rule as parameters

All parameters are required but may be empty length strings

Reads in the Snort Display configuration file to fill in each structure

Node structures represent the types of rules and contain

The name of the node

The number of protocols associated with the node

The number of rules associated with the node

The number of options associated with the node

A pointer to the next node

A pointer to a list of protocols associated with the node

List structures represent the types of network protocols and contain

The name

The number of rules

The number of options

A pointer to the next protocol

A pointer to a list of rules

Rule structures represent the header information of each rule and contain

The name

The number of options

A pointer to the next rule

A pointer to a list of options

Option structures represent the options section of each rule and contain

The name

The message that will be sent upon the option being matched

A pointer to the next option.

3.4.1.2. Snort functions

Function: ***CreateDefaultRules()***

Creates the five default link lists, one for each type of rule

Each link list is headed by a ListHead Snort data structure

Each ListHead contains a pointer to a list of RuleTreeNodes

Each element of the list contains unique header information supplied by the rules

Contained in each RuleTreeNode is a pointer to a list of OptTreeNodes

Each element of the list contains the options section of similar rules

Function: ***ParseRulesFile()***

Parses each rule file in the order it is referenced in the snort configuration file

3.4.1.3. Description of implementation

Two frames were created to split the user interface, the top frame, from the display area, the bottom frame. ***Menu.pl()***, generated the code to create the user interface. Since there is no Snort Display configuration file available for the first iteration, the default interface is displayed. Once a user selects "Browse", ***browse.pl()*** will open a new browser window and generate the HTML code to display the change-rooted directory. After a Snort configuration file is selected, JavaScript code will enter the full path of the file into the interface. Upon selecting "Load", ***config()*** is called to create the configuration file for Snort Display. ***Config()*** was wrapped by another script ***config.pl()*** to create the ***config()*** command line string. ***Config.pl()*** completes by calling ***menu.pl()*** which will read in the Snort Display configuration file that was created in the previous step. Included in the HTML code produced by ***menu.pl()*** are three drop down boxes

and JavaScript to dynamically submit the form. Each time a user makes a selection, ***MenuSelect()*** will verify the selection(s) made, possibly reset the remaining drop down boxes, and determine whether to call ***menu.pl()*** or ***display.pl()***.

Config() first initializes the environment by calling ***InitNetmasks()***, ***InitPreprocessors()***, ***InitPlugIns()***, and ***InitTag()***. ***InitVariables()*** is then called to set other variables to complete the initialization process. Many different variations of code were revised on a trial-and-error approach to determine which internal structures were required by the Snort functions to operate properly. ***CreateDefaultRules()*** and ***ParseRulesFile()*** are then called to read in the rule set. ***ParseRulesFile()*** will read in each rule in a first come, first serve basis. As each rule is read in, it will be concatenated to the link list it is associated with. By using some basic intelligence, any common rules will be combined to decrease the number of rules that need to be processed. For example, Rule 1 matches a range of addresses and a single port.

alert tcp any any -> 192.168.1.0/24 111

Rule 2 matches a single address, contained in the original range, and the same port.

alert tcp any any -> 192.168.1.10/32 111

Snort will flip-flop the two rules, otherwise, rule 2 would never be matched. This is because any traffic matching Rule 2 will match Rule 1 first. Finally, the Snort Display configuration file is created by ***CreateDataFile()***. Initially, the type of rule, protocol, rule number, option number, IP addresses, and port numbers were dumped from the link lists. Throughout the development process, additional properties were added which includes the signature id, the signature revision, and the message that will be sent to the logging and alerting mechanism.

Display() is the final script that is activated once the selection criteria has been made. Based on the information provided from the user and the Snort Display configuration file,

display() is able to create the graph. First, ***readFile()*** reads in the Snort Display configuration file. Next, ***drawImage()*** is called to create the graphic which references ***drawNodes()***, ***drawLists()***, ***drawRules()***, and ***drawOptions()*** to display each particular portion of the graph. Based on the depth of the selection criteria determines which of the functions will be called. Once the image is written to a file, ***freeImage()*** free any memory required to store the graph.

3.4.2. Final implementation

3.4.2.1. Snort Display functions

snortDisplay.pl()

Perl script

Calls ***browse.pl()***, ***config()*** and ***display()***

Became the interface into the application

Contains the functionality of ***config.pl()*** and ***display.pl()***

Display()

Receives the following additional fields via the list of command line parameters

The number of rules to display

Location of the selected rule, if any

The width of the screen

Added the following to the Node structure

The maximum height of the tree below the node

Added the following to the List structure

The maximum height of the tree below the protocol point on the graph

Added the following to the Rule structure

The source and destination IP address

The source and destination port number

A pointer to the previous rule in the list

Added the following to the Option structure

The id and revision number of the option

3.4.2.2. Description of implementation

After testing, it was determined that there was a problem updating the interface in certain scenarios. For example, if a node, protocol, and rule were selected, and then another protocol was selected, the rules were not updated correctly. The problem was that the decision to use JavaScript to create an interactive interface prevented the user interface from receiving the correct information based on the new selection. Therefore, the JavaScript interface was converted to a Perl script, ***snortDisplay.pl***, that produces HTML code to create a single frame containing the same information and functionality as the first generation of code. ***SnortDisplay.pl*** is now the single entry point for the entire application. Once the processing was moved from the client side to server side, the drop down select box fields were filled in correctly based on the user selections which provided a smoother user interface. The functionality provided by ***config.pl()*** was also added to ***snortDisplay.pl()*** to eliminate the script entirely. It was also determined later that the server side implementation provided a much easier implementation of the added functionality of displaying a portion of the graph.

After providing a demonstration of the first generation of code to the seminar group I was a part of, it was suggested that the ability to display a range of rules would provide better

functionality. Therefore, two more fields were added to the interface, a zoom and location field. Once a user has narrowed the search to a specific type and protocol, the two fields will appear. Now the user can specify how many rules will appear in the graph and the location of the selected rule. Once the complete selection is made, the form data is passed to ***display()*** which will dynamically create and display the graph. (Figure 4)

In order to display a range of rules, a few changes were made. First, a pointer was added to each internal data structure, that is represented in the link lists, that points to the previous element in the list. Now the link lists can be traversed in reverse, and therefore, a portion of the graph can be displayed. Finally, the width of the screen is passed to the display C program to limit the amount of space the graphic needs to cover. By limiting the size of the graphic file to the size of the screen will decrease the load time of the graphic file.

3.5. Final results

The final implementation provides the information that has been requested from the Snort user's group. ***SnortDisplay()*** provides the interface to the application to select a Snort configuration file and criteria to create the graph. Once a Snort configuration file is selected, ***config()*** retrieves the information from the link lists provided by Snort. ***Display()*** then uses that data and the user selection(s) to create a graphical representation of how Snort will interpret the rule set. The information provided by the graph creates a clearer picture of how Snort interprets the rule set. Thus, it assists a user to configure Snort to execute more efficiently.

CHAPTER 4

CONCLUSION AND FUTURE WORK

Snort Display is a great building block for a unique performance-tuning tool. As stated earlier, there is no mechanism to display how Snort processes the rule set. The rules files are self-explanatory but the extreme number of rules hinders a user from gathering a detailed understanding of the entire set. Documentation exists that explains the structure of the link lists, but the contents of the structures and how they are arranged are instance specific. Therefore, Snort Display fills in the missing gaps that exist to gain a better understanding of how Snort processes the rule set. With the information provided by Snort Display and a good understanding of the traffic that Snort will be processing, one can configure Snort for better performance.

Snort Display is a first attempt at providing an easy interface to describing how Snort processes the rule set provided. Fortunately, the majority of the components created, that interface with Snort, are dynamically configured based on how Snort is configured. However, this is an area that can be improved. For instance, the default rule types are the only types that are parsed out of the rule set. A dynamic implementation of this section will allow a user to change Snort, and in turn, change Snort Display without any additional changes. The graph size is also, somewhat, statically determined. Due to a lack of time, the max number of options associated with a rule determines the height of the graph regardless of the rule(s) selected. If the max number of options displayed determined the height of the image, the image size and load

time would be greatly reduced. Finally, the overall detail of the graph itself needs to be improved upon. A redesign of the tree may provide more space to include more detail on the graph. By decoding additional information contained in various structures, more detail will be available upon request. Snort Display can now provide the basis for other similar tools.

APPENDIX A

A.1. README

```
README file for Snort Display
------------------------------------------------------------

Requirements
------------
The following must be previously installed before configuring
and installing Snort Display.

                  Snort: http://www.snort.org/dl/
      GD Graphics Library: http://www.boutell.com/gd/

Installation Guide
------------------
1)  ./configure.pl <options>
      Configuration options:
            --snort | -s            Path to snort source code
            --snort_display | -d    Path to snort display source code
            --libs                  Path to linker libraries
            --tempdir | -t          Temporary directory
            --browsedir | -b        Base browse directory
            --datafile | -f         Path to data file
            --logfile | -l          Path to log file
            --imagefile             Full path of image file
            --imageuri              Virtual path of image file
            --html_path             Path to html directory
            --html_virtual          Virtual path to html files
            --cgi_path              Path to cgi-bin
            --cgi_virtual           Virtual path to cgi-bin
            --width                 Width of nodes
            --height                Height of nodes
            --space                 Space between nodes
            --help | -h             Display help information
2)  make
3)  make install


Usage
-----
The first step in using Snort Display is to provide the location of a Snort
configuration file.  The full path can be entered directly by hand, or, by
pressing the "Browse" button, another window will open displaying what is
considered the root path that was provided during installation.  The
remaining directory tree can be searched for a given Snort configuration
file.  Once selected, the full path will be entered into the interface.
```

```
Next, by pressing the "Load" button, the Snort configuration file will be
parsed and the required information will be written to a temparary file
located in the directory provided during installation.

Once the data file is loaded, the interface will initialize itself with the
default rule types read in from the data file.  By selecting various options
displayed in each drop down box, a graph will be created displaying the
requested information.  A graph can contain as little as a single rule, or,
as much as the entire file.


Questions / Comments
--------------------
Snort display is a first attempt at providing information about how Snort
parses the configuration file.  You may use the software as you see fit, that
being, copying the software, changing the software, and/or re-distributing
the software.  If you have any questions or comments on how to use the
software, or enhance the software, please send them to belans@cs.fsu.edu.
```

A.2. configure.pl

```
#!/usr/bin/perl

############################################################
# Copyright 2003 Joseph Belans <belans@cs.fsu.edu>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
############################################################
use strict;
use Getopt::Long;

# Set default values
my $GCC = `which gcc 2>/dev/null`;
chop ($GCC);
my $PERL = `which perl 2>/dev/null`;
chop ($PERL);
my $SNORT = "/usr/local/src";
$SNORT = `ls -d ${SNORT}/snort* 2>/dev/null | tail -1`;
chop ($SNORT);
my $SNORT_DISPLAY = `pwd`;
chop ($SNORT_DISPLAY);
${SNORT_DISPLAY} =~ s/ /\\ /g;
```

```
my $LIBS = "/usr/local/lib:/usr/lib";
my $TEMPDIR = "/tmp";
my $BROWSEDIR = "/usr/local/src";
my $DATAFILE = "${TEMPDIR}/data.txt";
my $LOGFILE = "${TEMPDIR}/log.txt";
my $HTML_PATH = "/var/www/html/snort_display";
my $VIRTUAL_HTML_PATH = "/snort_display";
my $CGI_PATH = "/var/www/cgi-bin/snort_display";
my $VIRTUAL_CGI_PATH = "/cgi-bin/snort_display";
my $IMAGEFILE = "$HTML_PATH/graph.jpg";
my $IMAGEURI = "$VIRTUAL_HTML_PATH/graph.jpg";
my $WIDTH = 5;
my $HEIGHT = 5;
my $SPACE = 10;
my $HELP;

# Parse command line parameters
my $result = GetOptions ("snort|s:s" => \$SNORT,
                  "snort_display|d:s" => \$SNORT_DISPLAY,
                  "libs:s" => \$LIBS,
                  "tempdir|t:s" => \$TEMPDIR,
                  "browsedir|b:s" => \$BROWSEDIR,
                  "datafile|f:s" => \$DATAFILE,
                  "logfile|l:s" => \$LOGFILE,
                  "imagefile:s" => \$IMAGEFILE,
                  "imageuri:s" => \$IMAGEURI,
                  "html_path:s" => \$HTML_PATH,
                  "html_virtual:s" => \$VIRTUAL_HTML_PATH,
                  "cgi_path:s" => \$CGI_PATH,
                  "cgi_virtual:s" => \$VIRTUAL_CGI_PATH,
                  "width:i" => \$WIDTH,
                  "height:i" => \$HEIGHT,
                  "space:i" => \$SPACE,
                  "help|h" => \$HELP);

# Check options that were passed
if ((!$result) || ($HELP)) {
   printUsage ();
   exit;
}

# Check that required components exist
# Find snort
if ( ! -f "$SNORT/src/snort.c" ) {
   print "Can not find snort source code.\n";
   exit;
}

# Check for gcc
if ($GCC eq "") {
   print "Can not find gcc compiler.\n";
   exit;
}

# Check for perl
if ($PERL eq "") {
   print "Can not find perl interpreter.\n";
```

```
   exit;
}

# Check third party libraries
my $gdflag = 0;
my $jpegflag = 0;
my $zlibflag = 0;
my $pngflag = 0;
my $freetypeflag = 0;
foreach (split (/:/, $LIBS)) {
   if (-f "$_/libgd.a") {
      $gdflag = 1;
   }
   if (-f "$_/libjpeg.a") {
      $jpegflag = 1;
   }
   if (-f "$_/libz.a") {
      $zlibflag = 1;
   }
   if (-f "$_/libpng.a") {
      $pngflag = 1;
   }
   if (`ls $_/libfreetype.* 2>/dev/null` ne "") {
      $freetypeflag = 1;
   }
}
if (!$gdflag) {
   print "Missing gd library.\n";
   exit;
}
if (!$jpegflag) {
   print "Missing jpeg library.\n";
   exit;
}
if (!$zlibflag) {
   print "Missing z library.\n";
   exit;
}
if (!$pngflag) {
   print "Missing png library.\n";
   exit;
}
if (!$freetypeflag) {
   print "Missing freetype library.\n";
   exit;
}

# Display parameters
print "Snort Source Code:\t\t\t$SNORT\n";
print "Snort Display Source Code:\t\t$SNORT_DISPLAY\n";
print "Temp Directory:\t\t\t\t$TEMPDIR\n";
print "Data File:\t\t\t\t$DATAFILE\n";
print "Log File:\t\t\t\t$LOGFILE\n";
print "Snort Display HTML Path:\t\t$HTML_PATH\n";
print "Snort Display Virtual HTML Path:\t$VIRTUAL_HTML_PATH\n";
print "Snort Display CGI Path:\t\t\t$CGI_PATH\n";
print "Snort Display Virtual CGI Path:\t\t$VIRTUAL_CGI_PATH\n";
```

```
# Rename main in snort.c
chdir "$SNORT/src";
system ("cp snort.c snort.org.c");
system ("cat snort.org.c | sed
's/\\(int.*\\)main\\(.*arg[cv].*arg[cv].*\\)/\\1mainSnort\\2/' > snort.c");

# Get command line to create config.c
my ($gcc_config) = `make -n snort.o`;
my ($gcc_snort) = `make -n snort | grep gcc | grep snort.o`;

# Create command lines
$gcc_config =~ s/\bsnort.c\b/config.c/;
$gcc_config =~ s/-I\.\./-I\$\(SNORT\)/g;
$gcc_config =~ s/-I\./-I\$\(SNORT\)\/src/g;

${gcc_snort} =~ s/\bsnort.o\b//;
${gcc_snort} =~ s/\b([a-zA-Z\-\_\/]+\.[o|a])\b/\$\(SNORT\)\/src\/\1/g;
${gcc_snort} =~ s/\bsnort\b/config config.o getcgi.o snort.o/;

# Create snortDisplay.html
chdir "$SNORT_DISPLAY";
my ($path_escaped) = $VIRTUAL_CGI_PATH;
$path_escaped =~ s/\//\\\//g;
my ($cmd) = "cat perl/snortDisplay.html.in | sed
's/<VIRTUAL_CGI_PATH>/$path_escaped/' > perl/snortDisplay.html";
system ("$cmd");

# Create snortDisplay.pl
$path_escaped = $PERL;
$path_escaped =~ s/\//\\\//g;
$cmd = "cat perl/snortDisplay.pl.in | sed 's/<PERL>/$path_escaped/' >
perl/snortDisplay.pl";
system ("$cmd");

# Create browse.pl
$cmd = "cat perl/browse.pl.in | sed 's/<PERL>/$path_escaped/' >
perl/browse.pl";
system ("$cmd");

# Create c makefile
open (OUTFILE, ">c/Makefile");
print OUTFILE "SNORT=$SNORT\n";
print OUTFILE "SNORT_DISPLAY=$SNORT_DISPLAY\n\n";
print OUTFILE "all: config display\n";
print OUTFILE "install: install-config install-display\n\n";
print OUTFILE "clean:\n";
print OUTFILE "\trm -f config display *.o header.h Makefile\n\n";
print OUTFILE "config: config.o getcgi.o\n";
print OUTFILE "\t$gcc_snort\n\n";
print OUTFILE "display: display.c getcgi.o displayFunctions.o\n";
print OUTFILE "\tgcc -o display display.c displayFunctions.o getcgi.o -lgd -
ljpeg";
foreach (split (/:/, $LIBS)) {
   print OUTFILE " -L$_";
}
```

```
print OUTFILE "\n\n";
print OUTFILE "config.o: header.h config.c\n";
print OUTFILE "\t$gcc_config\n\n";
print OUTFILE "getcgi.o: getcgi.h getcgi.c\n";
print OUTFILE "\tgcc -c getcgi.c\n\n";
print OUTFILE "displayFunctions.o: displayFunctions.h displayFunctions.c\n";
print OUTFILE "\tgcc -c displayFunctions.c\n\n";
print OUTFILE "install-config: config\n";
print OUTFILE "\tmkdir -p $CGI_PATH; \\\n";
print OUTFILE "\tcp -p config $CGI_PATH\n\n";
print OUTFILE "install-display: display\n";
print OUTFILE "\tmkdir -p $CGI_PATH; \\\n";
print OUTFILE "\tcp -p display $CGI_PATH\n";
close (OUTFILE);

# Create the perl Makefile
open (OUTFILE, ">perl/Makefile");
print OUTFILE "all: install\n\n";
print OUTFILE "install:\n";
print OUTFILE "\tcp *.pl $CGI_PATH; \\\n";
print OUTFILE "\tchmod 755 $CGI_PATH/*.pl; \\\n";
print OUTFILE "\tcp *.html $CGI_PATH; \\\n";
print OUTFILE "\tmkdir -p $HTML_PATH; \\\n";
print OUTFILE "\ttouch $IMAGEFILE; \\\n";
print OUTFILE "\tchmod 666 $IMAGEFILE\n\n";
print OUTFILE "clean:\n";
print OUTFILE "\trm browse.pl header.pl snortDisplay.pl snortDisplay.html
Makefile\n";
close (OUTFILE);

# Create snort
chdir "$SNORT/src";
system ("make snort.o");
system ("mv snort.o $SNORT_DISPLAY/c");
system ("cp snort.org.c snort.c");
system ("rm snort.org.c");
system ("make snort.o");
system ("make snort");

# Create c/header.h
chdir "$SNORT_DISPLAY";
open (OUTFILE, ">c/header.h");
print OUTFILE "#ifndef _snortDisplay\n\n";
print OUTFILE "/* Declare snortDisplay parameters */\n";
print OUTFILE "#define _snortDisplay\n\n";
print OUTFILE "/* Configuration section */\n";
print OUTFILE "#define TEMPDIR\t\"$TEMPDIR\"\n";
print OUTFILE "#define BASECGI\t\"$VIRTUAL_CGI_PATH\"\n";
print OUTFILE "#define WIDTH\t$WIDTH\n";
print OUTFILE "#define HEIGHT\t$HEIGHT\n";
print OUTFILE "#define SPACE\t$SPACE\n\n";
print OUTFILE "/* Create constants */\n";
print OUTFILE "#define DATAFILE\t\"$DATAFILE\"\n";
print OUTFILE "#define LOGFILE\t\t\"$LOGFILE\"\n";
print OUTFILE "#define IMAGEFILE\t\"$IMAGEFILE\"\n";
print OUTFILE "#define IMAGEURI\t\"$IMAGEURI\"\n\n";
print OUTFILE "#endif\n";
```

```
close (OUTFILE);

# Create perl/header.pl
open (INFILE, "perl/header.pl.in");
open (OUTFILE, ">perl/header.pl");
foreach (<INFILE>) {
   chop;
   if ($_ eq "<INSERT>") {
      print OUTFILE "# Constants\n";
      print OUTFILE "use constant BASECONFDIR => \"$BROWSEDIR\";\n";
      print OUTFILE "use constant DATAFILE => \"$DATAFILE\";\n";
      print OUTFILE "use constant TEMPLATE =>
\"$CGI_PATH/snortDisplay.html\";\n";
      print OUTFILE "use constant BASECGIDIR => \"$CGI_PATH\";\n\n";
      print OUTFILE "use constant BASECGIURI => \"$VIRTUAL_CGI_PATH\";\n\n";
   } else {
      print OUTFILE "$_\n";
   }
}
close (OUTFILE);

#########################################################################
# Function: printUsage
#
# Parameters: none
#
# Return: none
#
# Display the usage of configure
#
sub printUsage () {
   print "configure <options>\n";
   print "\t--snort | -s\t\tPath to snort source code\n";
   print "\t--snort_display | -d\tPath to snort display source code\n";
   print "\t--libs\t\t\tPath to linker libraries\n";
   print "\t--tempdir | -t\t\tTemporary directory\n";
   print "\t--browsedir | -b\tBase browse directory\n";
   print "\t--datafile | -f\t\tPath to data file\n";
   print "\t--logfile | -l\t\tPath to log file\n";
   print "\t--imagefile\t\tFull path of image file\n";
   print "\t--imageuri\t\tVirtual path of image file\n";
   print "\t--html_path\t\tPath to html directory\n";
   print "\t--html_virtual\t\tVirtual path to html files\n";
   print "\t--cgi_path\t\tPath to cgi-bin\n";
   print "\t--cgi_virtual\t\tVirtual path to cgi-bin\n";
   print "\t--width\t\t\tWidth of nodes\n";
   print "\t--height\t\tHeight of nodes\n";
   print "\t--space\t\t\tSpace between nodes\n";
   print "\t--help | -h\t\tDisplay help information\n";
}
```

A.3. Makefile

```
all:
      cd c; \
      make

install:
      cd c; \
      make install; \
      cd ../perl; \
      make install

clean:
      cd c; \
      make clean; \
      cd ../perl; \
      make clean
```

A.4. config.c

```
/*************************************************************
 * Copyright 2003 Joseph Belans <belans@cs.fsu.edu>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *************************************************************/
/*************************************************************
 * File: config.c
 *
 * Parameters:
 *    configuration file
 *    output file
 *
 * Returns: none
 *
 * Dump the link lists provided by Snort to a text file
 *************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "header.h"
#include "getcgi.h"
```

```
#include "snort.h"
#include "rules.h"
#include "strlcpyu.h"
#include "plugbase.h"
#include "parser.h"
#include "tag.h"
#include "util.h"

/* Start of the rule list */
extern RuleListNode *RuleLists;

/* Declare functions */
void InitVariables ();
void StoreConfig (char *file);
void CreateDataFile (char *outfile);
void ProcessNode (FILE *outfile, RuleListNode *node);
void ProcessList (FILE *outfile, RuleTreeNode *list);
void ConvertIP (IpAddrSet *idx, char **str);

/* Main function */
int main(int argc, char* argv[]) {
   char     *outfile,         /* Output file */
      *infile;          /* Config file */

   /* Use stdout for output */
   if (argc == 2) {
      outfile = NULL;
      infile = argv[1];

   /* Set input and output file */
   } else if (argc == 3) {
      outfile = DATAFILE;
      infile = argv[1];

   /* Display usage */
   } else {
      printf ("USAGE: config <config file> [output file]\n");
      return (0);
   }

   /* Setup some lookup data structs */
   InitNetmasks();

   /* Initialize required variables */
   InitVariables ();

   /* Store the config file */
   StoreConfig (infile);

   /* Initialize all the plugin modules */
   InitPreprocessors();
   InitPlugIns();
   InitTag();

   /* Setup the default rule action anchor points */
   CreateDefaultRules();
```

```
   /* Parse the rules file */
   ParseRulesFile(pv.config_file, 0);

   /* Create the data file */
   CreateDataFile (outfile);

   /* Done */
   return (0);
}

/*****************************************************************
 * Function: InitVariables
 *
 * Parameters: none
 *
 * Returns: (void)
 *
 * Sets the required variables
 *****************************************************************/
void InitVariables () {
   /* Eliminate any messages */
   pv.verbose_flag = 0;
   pv.quiet_flag = 1;
   pv.nolog_flag = 1;
   pv.log_cmd_override = 1;
   pv.alert_mode = ALERT_NONE;

   /* Set the alert filename */
   pv.alert_filename = LOGFILE;

   /* Set the default alert mode */
   pv.alert_mode = ALERT_FULL;

   /* Set the default assurance mode (used with stream 4) */
   pv.assurance_mode = ASSURE_ALL;

   pv.use_utc = 0;

   /* Turn off decoder alerts */
   pv.decode_alert_flag = 0;

   /* Set the default logging directory */
   strlcpy(pv.log_dir, TEMPDIR, STD_BUF);
}

/*****************************************************************
 * Function: StoreConfig
 *
 * Parameters:
 *    file (char*) - the output config file name
 *
 * Returns: (void)
 *
 * Stores the config file name in the pv structure
 *****************************************************************/
void StoreConfig (char *file) {
   char *tmp;
```

```
   strlcpy(pv.config_file, file, STD_BUF);
   if (strrchr(file,'/')) {
      strlcpy(pv.config_dir, file, STD_BUF);
      tmp = strrchr(pv.config_dir,'/');
      *(++tmp) = '\0';
   } else {
      strlcpy(pv.config_dir, "./", STD_BUF);
   }

   pv.use_rules = 1;
}

/*****************************************************************
 * Function: CreateDataFile
 *
 * Parameters:
 *    outfile (char*) - the output config file name
 *
 * Returns: (void)
 *
 * Creates the data file that contains the rule set
 *****************************************************************/
void CreateDataFile (char *outfile) {
   FILE *outfile_handle;      /* Output file handle */
   RuleListNode *node;          /* Current node */
   int i;

   /* Output to stdout */
   if (!outfile) {
      outfile_handle = stdout;

   /* Open output file */
   } else if (!(outfile_handle = fopen (outfile, "w"))) {
      printf ("Can't open output file: %s\n", outfile);
      return;
   }

   /* Write the config file name */
   fprintf (outfile_handle, "%s\n", pv.config_file);

   node = RuleLists;            /* The first node */

   /* Loop through all nodes */
   for (i = 0; node; i++) {
      /* If the current node has rules, process */
      if (node->RuleList) {
       /* Write the node name */
       fprintf (outfile_handle, "%s\n", node->name);

         /* Process the lists associated with the current node */
         ProcessNode (outfile_handle, node);
      }

      node = node->next;
   }
```

```
   /* Close the output file, if necessary */
   if (outfile)
      fclose (outfile_handle);
}

/*****************************************************************
 * Function: ProcessNode
 *
 * Parameters:
 *    outfile (FILE*) - the output config file handle
 *    node (RuleListNode*) - the head node
 *
 * Returns: (void)
 *
 * Parses the link list for the given node to collect the required data
 * from each protocol list associated with the node
 *****************************************************************/
void ProcessNode (FILE *outfile, RuleListNode *node) {
   /* Process the lists for the node */
   if (node->RuleList->IpList) {
      fprintf (outfile, " Ip\n");
      ProcessList (outfile, node->RuleList->IpList);
   }
   if (node->RuleList->TcpList) {
      fprintf (outfile, " Tcp\n");
      ProcessList (outfile, node->RuleList->TcpList);
   }
   if (node->RuleList->UdpList) {
      fprintf (outfile, " Udp\n");
      ProcessList (outfile, node->RuleList->UdpList);
   }
   if (node->RuleList->IcmpList) {
      fprintf (outfile, " Icmp\n");
      ProcessList (outfile, node->RuleList->IcmpList);
   }
}

/*****************************************************************
 * Function: ProcessList
 *
 * Parameters:
 *    outfile (FILE*) - the output config file handle
 *    node (RuleTreeNode*) - the head of the list
 *
 * Returns: (void)
 *
 * Parses the link list for the given node to collect the required data
 * from each protocol list associated with the node
 *****************************************************************/
void ProcessList (FILE *outfile, RuleTreeNode *list) {
   RuleTreeNode *rule;              /* Current rule */
   OptTreeNode *option;             /* Current option */
   IpAddrSet *idx;                  /* Indexing pointer */
   char *str = (char*)malloc (42);  /* Tmp string */

   /* Get the first rule */
   rule = list;
```

```
/* Loop through the list */
while (rule) {
   fprintf (outfile, "  Rule %d:", rule->head_node_number);

   /* Display port information */
   if (rule->flags & EXCEPT_SRC_PORT)
    fprintf (outfile, "!");

   if (rule->flags & ANY_SRC_PORT)
    fprintf (outfile, "any:");
   else
    fprintf (outfile, "%d/%d:", rule->hsp, rule->lsp);

   if (rule->flags & EXCEPT_DST_PORT)
    fprintf (outfile, "!");

   if (rule->flags & ANY_DST_PORT)
    fprintf (outfile, "any:(");
   else
    fprintf (outfile, "%d/%d:(", rule->hdp, rule->ldp);

   /* Print source ip info */
   idx = rule->sip;
   while(idx != NULL) {
    /* Display ip address info */
    ConvertIP (idx, &str);
    fprintf (outfile, "%s", str);

    /* Get the next address */
    idx = idx->next;

    /* Print separator */
    if (idx)
       fprintf (outfile, ":");
   }
   fprintf (outfile, "):(");

   /* Print destination ip */
   idx = rule->dip;
   while(idx != NULL) {
    /* Display ip address info */
    ConvertIP (idx, &str);
    fprintf (outfile, "%s", str);

    /* Get the next address */
    idx = idx->next;

    /* Print separator */
    if (idx)
       fprintf (outfile, ":");
   }
   fprintf (outfile, ")\n");

   /* Loop through the options */
   option = rule->down;
   while (option) {
```

```
         fprintf (outfile, "   Option %d:%d:%d:%s\n",
            option->chain_node_number, option->sigInfo.id,
            option->sigInfo.rev, option->sigInfo.message);
         option = option->next;
      }

      rule = rule->right;
   }

   free (str);
}

void ConvertIP (IpAddrSet *idx, char **str) {
   u_int8_t octet;
   u_int32_t tmp;
   char *strptr;
   int i;

   /* Set pointer to string */
   strptr = *str;

   /* Check if the exception flag is set */
   if (idx->addr_flags & EXCEPT_IP) {
      sprintf (strptr, "!");
      strptr++;
      memset (strptr, '\0', 1);
   }

   /* Check if this is an "any" ip */
   if ((idx->ip_addr == 0) && (idx->netmask == 0)) {
      sprintf (strptr, "any");
      return;
   }

   /* Convert IP address */
   tmp = idx->ip_addr;
   for (i = 0; i < 4; i++) {
      octet = tmp;
      sprintf (strptr, "%d.", octet);
      tmp = tmp >> 8;
      strptr = strrchr (*str, '\0');
   }
   memset ((strptr - 1), '/', 1);

   /* Convert netmask */
   tmp = idx->netmask;
   for (i = 0; i < 4; i++) {
      octet = tmp;
      sprintf (strptr, "%d.", octet);
      tmp = tmp >> 8;
      strptr = strrchr (*str, '\0');
   }
   strptr--;
   memset (strptr, '\0', 1);
}
```

A.5. display.c

```
/*********************************************************
 * Copyright 2003 Joseph Belans <belans@cs.fsu.edu>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *********************************************************/
/*********************************************************
 * File: display.c
 *
 * Parameters:
 *      node selected
 *      list selected
 *    rule selected
 *    number of rules to display
 *    the available width of the screen
 *    location of the selected node
 *
 * Returns: none
 *
 * Displays the graph
 *********************************************************/
#include <stdio.h>
#include <string.h>
#include "header.h"
#include "displayFunctions.h"

/* Declare functions */
node *readFile (char *infile, int *numNodes,
                int *numLists, int *numRules, int *maxOptions);
void endHTML (int width, int height);

int main (int argc, char **argv) {
   node *root = NULL;      /* The root node */
   int numNodes = 0,      /* Number of nodes */
      numLists = 0,      /* Total number of lists */
      numRules = 0,      /* Total number of rules */
      maxOptions = 0;      /* Max number of options */

   /* Check if all arguments were supplied */
   if (argc != 7) {
      printf ("Must supply node, list, rule, zoom, width, "
                              "and location arguments<BR>\n");
```

```
      return (0);
   }

   /* Fill in the args structure */
   args.node = strdup (argv[1]);
   args.list = strdup (argv[2]);
   args.rule = strdup (argv[3]);
   args.zoom = atoi (argv[4]);
   args.width = atoi (argv[5]);

   if (strcasecmp (argv[6], "front") == 0)
      args.location = FRONT;
   else if (strcasecmp (argv[6], "middle") == 0)
      args.location = MIDDLE;
   else if (strcasecmp (argv[6], "end") == 0)
      args.location = END;
   else
      args.location = 0;

   /* Read in the data file */
   if (!(root = readFile (DATAFILE, &numNodes,
                  &numLists, &numRules, &maxOptions))) {
      printf ("Can't read data file: %s\n", DATAFILE);
      return (0);
   }

   /* Draw image */
   drawImage (root, maxOptions, IMAGEFILE);

   /* Free memory */
   freeImage (root);
   free (args.node);
   free (args.list);
   free (args.rule);

   /* Done */
   return (0);
}
```

```
/*****************************************************************
 * Function: readFile
 *
 * Parameters:
 *    infile (char*): data file
 *    numNodes (int*): number of nodes
 *    numLists (int*): number of lists
 *    numRules (int*): number of rules
 *    maxOptions (int*): max number of options
 *
 * Returns:
 *    Address to first node
 *    0 if there is an error
 *
 * Read in the data file and display the requested portion of the graph
 *****************************************************************/
node *readFile (char *infile, int *numNodes,
                        int *numLists, int *numRules, int *maxOptions) {
   node *root,          /* The root node */
      *curNode;   /* The current node */
   list *curList; /* The current list */
   rule *curRule; /* The current rule */
   option *curOption;   /* The current option */
   FILE *infile_handle; /* Infile pointer */
   char str[2056],      /* Temp string */
      *tmp;

   /* Open the data file */
   if (!(infile_handle = fopen (infile, "r")))
      return (0);

   /* Get config file name */
   fgets (str, 2056, infile_handle);

   /* Read until EOF */
   root = (node*)malloc (sizeof (node));
   root->head = NULL;
   curNode = root;
   while ((fgets (str, 2056, infile_handle))) {
      /* Strip off the new line */
      memset (strchr(str, '\n'), '\0', 1);

      /* Increment / Reset counters */
      /* Current line contains an option */
      if (str[2] == ' ') {
       /* Create a new option */
       if (curRule->numOptions == 0) {
          curRule->head = (option*)malloc (sizeof (option));
          curOption = curRule->head;
       } else {
          curOption->next = (option*)malloc (sizeof (option));
          curOption = curOption->next;
       }

       /* Fill in option structure */
       /* Set the name */
       tmp = &str[3];
```

```
 memset (strchr (tmp, ':'), '\0', 1);
 curOption->name = strdup (tmp);

 /* Set the id */
 tmp += strlen (curOption->name) + 1;
 memset (strchr (tmp, ':'), '\0', 1);
 curOption->id = atoi (tmp);

 /* Set the revision number */
 tmp += strlen (tmp) + 1;
 memset (strchr (tmp, ':'), '\0', 1);
 curOption->rev = atoi (tmp);

 /* Set the msg string */
 tmp += strlen (tmp) + 1;
 curOption->msg = strdup (tmp);

 /* Set the next pointer */
 curOption->next = NULL;

 /* Set option counters */
 curRule->numOptions++;
 curList->numOptions++;
 curNode->numOptions++;

 /* Set new maxOptions if necessary */
 if ((*maxOptions) < curRule->numOptions)
    (*maxOptions) = curRule->numOptions;

/* Current line contains a rule */
} else if (str[1] == ' ') {
 /*  Create a new rule */
 if (curList->numRules == 0) {
    curList->head = (rule*)malloc (sizeof (rule));
    curList->maxOptions = 0;
    curRule = curList->head;
    curRule->prev = NULL;
 } else {
    if (curRule->numOptions > curList->maxOptions)
      curList->maxOptions = curRule->numOptions;

    curRule->next = (rule*)malloc (sizeof (rule));
    (curRule->next)->prev = curRule;
    curRule = curRule->next;
 }

 /* Fill in the rule structure */
 /* Set the rule name */
 tmp = &str[2];
 memset (strchr (tmp, ':'), '\0', 1);
 curRule->name = strdup (tmp);

 /* Set the source port */
 tmp += strlen (curRule->name) + 1;
 memset (strchr (tmp, ':'), '\0', 1);
 curRule->src_port = strdup (tmp);
```

```
 /* Set the destination port */
 tmp += strlen (curRule->src_port) + 1;
 memset (strchr (tmp, ':'), '\0', 1);
 curRule->dst_port = strdup (tmp);

 /* Set the source ip address */
 tmp += (strlen (curRule->dst_port) + 2);
 memset (strchr (tmp, ')'), '\0', 1);
 curRule->src_ip = strdup (tmp);

 /* Set the destination ip address */
 tmp += (strlen (curRule->src_ip) + 3);
 memset (strchr (tmp, ')'), '\0', 1);
 curRule->dst_ip = strdup (tmp);

 /* Set the pointers */
 curRule->next = NULL;
 curRule->head = NULL;

 /* Set rule counters */
 curRule->numOptions = 0;
 curList->numRules++;
 curNode->numRules++;
 (*numRules)++;

/* Current line contains a list */
} else if (str[0] == ' ') {
 /* Create a new list */
 if (curNode->numLists == 0) {
    curNode->maxOptions = 0;
    curNode->head = (list*)malloc (sizeof (list));
    curList = curNode->head;
 } else {
    if (curList->maxOptions > curNode->maxOptions)
      curNode->maxOptions = curList->maxOptions;

    curList->next = (list*)malloc (sizeof (list));
    curList = curList->next;
 }

 /* Fill in the list structure */
 curList->name = strdup (&str[1]);
 curList->next = NULL;
 curList->head = NULL;

 /* Set list counters */
 curList->numRules = 0;
 curList->numOptions = 0;
 curNode->numLists++;
 (*numLists)++;

/* Current line contains a node */
} else {
 /* Create new node */
 if ((*numNodes) != 0) {
    curNode->next = (node*)malloc (sizeof (node));
    curNode = curNode->next;
```

```
        }
        curNode->name = strdup (str);

        /* Set node counters */
        (*numNodes)++;
        curNode->numLists = 0;
        curNode->numRules = 0;
        curNode->numOptions = 0;

        /* Set node pointers */
        curNode->next = NULL;
        curNode->head = NULL;
       }
   }

   /* Close infile */
   fclose (infile_handle);

   /* Return */
   return (root);
}

/*****************************************************************
 * Function: endHTML
 *
 * Parameters:
 *    width (int): width of image
 *    height (int): height of image
 *
 * Returns: void
 *
 * Display the end of the html code
 *****************************************************************/
void endHTML (int width, int height) {
   printf ("</MAP>\n"
   "<P><DIV style=\"z-index: 2\">\n"
   "<IMG SRC=\"%s\" WIDTH=%d HEIGHT=%d USEMAP=\"#mainmap\" BORDER=\"0\">\n"
   "</DIV>\n"
   "</BODY></HTML>\n", IMAGEURI, width, height);
}
```

A.6. displayFunctions.c

```
/*******************************************************
 * Copyright 2003 Joseph Belans <belans@cs.fsu.edu>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *******************************************************/
/*******************************************************
 * File: displayFunctions.c
 *
 * Functions to display the graph
 *******************************************************/
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <gd.h>
#include <gdfontl.h>
#include <gdfonts.h>
#include "displayFunctions.h"
#include "header.h"
#include "getcgi.h"

/* Declare globals */
int height = HEIGHT;
int width = WIDTH;
int space = SPACE;

/* Declare colors */
int black, red, green, blue;

/* Declare functions */
void displayPoint (gdImagePtr *im, int x, int y, char *node,
      char *list, char *rule, char *option, int color, char *fmt, ...);
void displayLine (gdImagePtr *im, void *head, void *next, int x1, int y1,
            int x2, int y2, int x3, int y3, int x4, int y4, int color);
char *createIPString (char *str);

/*************************************************************
 * Function: drawImage
 *
 * Parameters:
 *    root (node*): the root node
 *    maxOptions (int): max number of options
 *    file (char*): data file
 *
 * Returns: void
 *
 * Draw the graph
 *************************************************************/
void drawImage (node *root, int maxOptions, char *file) {
   node *curNode;             /* Node pointer */
   list *curList;             /* List pointer */
   rule *curRule;             /* Rule pointer */
   gdImagePtr im;             /* Image pointer */
   FILE *image;                     /* Image file */
   int white,                       /* White index */
      total_width = 0,        /* Width of image */
      total_height = 0;       /* Height of image */
```

```
/* Check if there is a root node */
if (!root)
   return;

/* Determine the size of the image */
curNode = root;

/* If all nodes, count all rules */
if (strcmp (args.node, "all") == 0)  {
   while (curNode != NULL) {
    total_width += curNode->numRules;

    if (curNode->maxOptions > total_height)
       total_height = curNode->maxOptions;

    curNode = curNode->next;
   }

} else {
   /* Find selected node */
   while (strcmp (args.node, curNode->name) != 0)
    curNode = curNode->next;

   curList = curNode->head;

   /* If all lists, count all lists */
   if (strcmp (args.list, "all") == 0)  {
    while (curList != NULL) {
       total_width += curList->numRules;

       if (curList->maxOptions > total_height)
         total_height = curList->maxOptions;

       curList = curList->next;
    }

   } else {
    /* Find selected list */
    while (strcmp (args.list, curList->name) != 0)
       curList = curList->next;

    /* If all rules, count all rules */
    if (strcmp (args.rule, "all") == 0) {
       total_width = curList->numRules;
       total_height = curList->maxOptions;

    } else {
       width *= 4;
       height *= 4;
       space *= 2;

       /* Determine height */
       if (args.zoom > 0) {
         total_width = args.zoom;
         total_height = curList->maxOptions;
       } else {
```

```
            total_width = args.width;

            curRule = curList->head;
            while (strcmp (args.rule, curRule->name) != 0)
               curRule = curRule->next;
            total_height = curRule->numOptions;
          }
       }
      }
   }

   /* Calculate total width and height */
   total_height = (total_height * (space + height)) +
                             (3 * (space + height)) + 120;

   if ((strcmp (args.rule, "") == 0) ||
            (strcmp (args.rule, "all") == 0) || (args.zoom > 0))
      total_width = (total_width * (space + width)) +
                             (3 * (space + width)) + 120;

   /* Create image background */
   im = gdImageCreate(total_width, total_height);

   /* Set background color */
   white = gdImageColorAllocate(im, 255, 255, 255);

   /* Declare colors */
   black = gdImageColorAllocate(im, 0, 0, 0);
   red = gdImageColorAllocate(im, 255, 0, 0);
   green = gdImageColorAllocate(im, 0, 255, 0);
   blue = gdImageColorAllocate(im, 0, 0, 255);

   /* Display the legend */
   gdImageString(im, gdFontLarge, 10, 10, "Legend:", black);
   gdImageFilledRectangle(im, 20, 30, 30, 40, black);
   gdImageString(im, gdFontLarge, 35, 30, "Type of rules", black);
   gdImageFilledRectangle(im, 20, 50, 30, 60, red);
   gdImageString(im, gdFontLarge, 35, 50, "Protocol of rule", red);
   gdImageFilledRectangle(im, 20, 70, 30, 80, green);
   gdImageString(im, gdFontLarge, 35, 70, "Rule", green);
   gdImageFilledRectangle(im, 20, 90, 30, 100, blue);
   gdImageString(im, gdFontLarge, 35, 90, "Options", blue);

   /* Create graph */
   drawNodes (&im, root, 20, 120);

   /* End html */
   endHTML (total_width, total_height);

   /* Write the image to a file */
   image = fopen(file, "wb");
   gdImageJpeg(im, image, -1);
   fclose(image);

   /* Destroy the image in memory. */
   gdImageDestroy(im);
}
```

```
/**************************************************************
 * Function: drawNodes
 *
 * Parameters:
 *    im (gdImagePtr*): pointer to image data
 *    head (node*): the head node
 *    x (int): x coordinate
 *    y (int): y coordinate
 *
 * Returns: void
 *
 * Draw the nodes
 **************************************************************/
void drawNodes (gdImagePtr *im, node *head, int x, int y) {
   node *curNode = head,      /* Head of node list */
      *stopNode = NULL, /* Stop node */
      *prevNode = NULL; /* Previous node */
   int x1 = x,                /* X coordinate */
      start, stop;            /* Start / Stop point of line */
   int color;                 /* Color of point */

   /* Set point color */
   color = gdImageColorAllocate (*im, 0, 0, 0);

   /* If not displaying All, find node */
   if (strcmp (args.node, "all") != 0)  {
      while (curNode != NULL) {
       if (strcmp (args.node, curNode->name) == 0)  {
          stopNode = curNode->next;
          curNode->next = NULL;
          break;
       }
       curNode = curNode->next;
      }
   }

   /* Display nodes */
   while (curNode != NULL) {
      /* Display point */
      displayPoint (im, x1, y, curNode->name, NULL, NULL, NULL, color,
            "<TABLE><TR><TD ALIGN=right>Name:</TD><TD>%s</TD></TR>\\n"
            "<TR><TD ALIGN=right>Lists:</TD><TD>%d</TD></TR>\\n"
            "<TR><TD ALIGN=right>Rules:</TD><TD>%d</TD></TR>\\n"
            "<TR><TD ALIGN=right>Options:</TD><TD>%d</TD></TR></TABLE>\\n",
            curNode->name, curNode->numLists,
            curNode->numRules, curNode->numOptions);

      /* Set starting point */
      start = x1;

      /* Draw lists */
      if (curNode->head)
       drawLists (im, curNode->head, curNode->name,
                              &x1, (y + height + space));
      else
       x1 += width + space;
```

```
      /* Set stopping point */
      stop = x1;

      /* Display line(s) */
      displayLine (im, curNode->head, curNode->next, (start + width),
      (y + (height / 2)), stop, (y + (height / 2)), (start + (width / 2)),
      (y + height), (start + (width / 2)), (y + height + space), color);

      /* Get next node */
      prevNode = curNode;
      curNode = curNode->next;
   }

   if (stopNode) {
      curNode = prevNode;
      curNode->next = stopNode;
   }
}

/*****************************************************************
 * Function: drawLists
 *
 * Parameters:
 *    im (gdImagePtr*): pointer to image data
 *    head (list*): the head list
 *    node (char*): name of the associated node
 *    x (int*): x coordinate
 *    y (int): y coordinate
 *
 * Returns: void
 *
 * Draw the lists
 *****************************************************************/
void drawLists (gdImagePtr *im, list *head, char *node, int *x, int y) {
   list *curList = head,      /* Head of protocol list */
      *stopList = NULL, /* Stop list */
      *prevList = NULL; /* Previous list */
   int x1 = *x,               /* X coordinate */
      start, stop;            /* Start / Stop point of line */
   int color;                 /* Color of point */

   /* Set point color */
   color = gdImageColorAllocate (*im, 255, 0, 0);

   /* If not displaying All, find list */
   if ((strcmp (args.list, "all") != 0) && (strcmp (args.list, "") != 0))  {
      while (curList != NULL) {
       if (strcmp (args.list, curList->name) == 0)  {
          stopList = curList->next;
          curList->next = NULL;
          break;
       }
       curList = curList->next;
      }
   }
```

```
   /* Display lists */
   while (curList != NULL) {
      /* Display point */
      displayPoint (im, x1, y, node, curList->name, NULL, NULL, color,
            "<TABLE><TR><TD ALIGN=right>Name:</TD><TD>%s</TD></TR>\\n"
            "<TR><TD ALIGN=right>Rules:</TD><TD>%d</TD></TR>\\n"
            "<TR><TD ALIGN=right>Options:</TD><TD>%d</TD></TR></TABLE>\\n",
            curList->name, curList->numRules, curList->numOptions);

      /* Set starting point */
      start = x1;

      /* Draw rules */
      if (curList->head)
       drawRules (im, curList->head,
                  node, curList->name, &x1, (y + height + space));
      else
       x1 += width + space;

      /* Set stopping point */
      stop = x1;

      /* Display line(s) */
      displayLine (im, curList->head, curList->next, (start + width),
      (y + (height / 2)), stop, (y + (height / 2)), (start + (width / 2)),
      (y + height), (start + (width / 2)), (y + height + space), color);

      /* Get next node */
      prevList = curList;
      curList = curList->next;
   }

   if (stopList) {
      curList = prevList;
      curList->next = stopList;
   }

   /* Return x coordinate */
   *x = x1;
}

/*************************************************************
 * Function: drawRules
 *
 * Parameters:
 *    im (gdImagePtr*): pointer to image data
 *    head (rule*): the head list
 *    node (char*): name of the associated node
 *    list (char*): name of the associated list
 *    x (int*): x coordinate
 *    y (int): y coordinate
 *
 * Returns: void
 *
 * Draw the rules
 *************************************************************/
void drawRules (gdImagePtr *im, rule *head,
```

```
               char *node, char *list, int *x, int y) {
rule *curRule = head,      /* Head of rule list */
   *startRule = NULL,      /* Start rule */
   *stopRule = NULL, /* Stopping point */
   *prevRule = NULL; /* Previous rule */
int x1 = *x,               /* X coordinate */
   start, stop,            /* Start / Stop point of line */
   color,                  /* Color of point */
   i;                /* Loop variables */
char *srcIP, *dstIP;       /* Src / Dst ip addresses */

/* Set point color */
color = gdImageColorAllocate (*im, 0, 255, 0);

/* If not displaying All, find rule */
if ((strcmp (args.rule, "all") != 0) && (strcmp (args.rule, "") != 0))  {
   while (curRule != NULL) {
    if (strcmp (args.rule, curRule->name) == 0)  {
       if (args.zoom == 0) {
         stopRule = curRule->next;
         curRule->next = NULL;
       }
       break;
    }
    curRule = curRule->next;
   }

   /* If zoom, find starting rule */
   if (args.zoom > 0) {
    if (args.location != FRONT) {
       if (args.location == MIDDLE)
         i = args.zoom / 2;
       else
         i = args.zoom - 1;

       for (; ((curRule->prev != NULL) && (i > 0)); i--)
         curRule = curRule->prev;
    }

    startRule = curRule;

    for (i = 1; i < args.zoom; i++)
       curRule = curRule->next;

    stopRule = curRule->next;
    curRule->next = NULL;
    curRule = startRule;
   }
}

/* Display rules */
while (curRule != NULL) {
   /* Create the temporary strings */
   if (strchr (curRule->src_ip, ':'))
    srcIP = createIPString (curRule->src_ip);
   else
    srcIP = strdup (curRule->src_ip);
```

```
      if (strchr (curRule->dst_ip, ':'))
       dstIP = createIPString (curRule->dst_ip);
      else
       dstIP = strdup (curRule->dst_ip);

      /* Display point */
      displayPoint (im, x1, y, node, list, curRule->name, NULL, color,
            "<TABLE><TR><TD ALIGN=right>Name:</TD><TD>%s</TD></TR>\\n"
            "<TR><TD ALIGN=right>Options:</TD><TD>%d</TD></TR>\\n"
            "<TR><TD ALIGN=right VALIGN=top>Src IP:</TD><TD>%s</TD></TR>\\n"
            "<TR><TD ALIGN=right>Src Port:</TD><TD>%s</TD></TR>\\n"
            "<TR><TD ALIGN=right VALIGN=top>Dest IP:</TD><TD>%s</TD></TR>\\n"
            "<TR><TD ALIGN=right>Dest Port:</TD><TD>%s</TD></TR></TABLE>\\n",
            curRule->name, curRule->numOptions, srcIP,
            curRule->src_port, dstIP, curRule->dst_port);

      /* Set starting point */
      start = x1;

      /* Draw options */
      if (curRule->head)
       drawOptions (im, curRule->head, &x1, (y + height + space));
      else
       x1 += width + space;

      /* Set stopping point */
      stop = x1;

      /* Display line(s) */
      displayLine (im, curRule->head, curRule->next, (start + width),
      (y + (height / 2)), stop, (y + (height / 2)), (start + (width / 2)),
      (y + height), (start + (width / 2)), (y + height + space), color);

      /* Get next node */
      prevRule = curRule;
      curRule = curRule->next;

      /* Free the temp strings */
      free (srcIP);
      free (dstIP);
   }

   if (stopRule) {
      curRule = prevRule;
      curRule->next = stopRule;
   }

   /* Return x coordinate */
   *x = x1;
}
```

```
/*****************************************************************
 * Function: drawOptions
 *
 * Parameters:
 *    im (gdImagePtr*): pointer to image data
 *    head (rule*): the head list
 *    x (int*): x coordinate
 *    y (int): y coordinate
 *
 * Returns: void
 *
 * Draw the options
 *****************************************************************/
void drawOptions (gdImagePtr *im, option *head, int *x, int y) {
   option *curOption = head;  /* Head of option list */
   int x1 = *x,               /* X coordinate */
      y1 = y;                 /* Y coordinate */
   int color;                 /* Color of point */

   /* Set point color */
   color = gdImageColorAllocate (*im, 0, 0, 255);

   /* Display options */
   while (curOption != NULL) {
      /* Display point */
      displayPoint (im, x1, y1, NULL, NULL, NULL, curOption->msg, color,
            "<TABLE><TR><TD ALIGN=right>Name:</TD><TD>%s</TD></TR>\\n"
            "<TR><TD ALIGN=right>Desc:</TD><TD>%s</TD></TR>\\n"
            "<TR><TD ALIGN=right>ID:</TD><TD>%d</TD></TR>\\n"
            "<TR><TD ALIGN=right>Rev:</TD><TD>%d</TD></TR></TABLE>\\n",
            curOption->name, curOption->msg, curOption->id, curOption->rev);

      /* Display line(s) */
      displayLine (im, NULL, curOption->next, (x1 + (width / 2)),
            (y1 + height), (x1 + (width / 2)), (y1 + height + space),
            0, 0, 0, 0, color);

      /* Get next point */
      y1 += height + space;

      /* Get next option */
      curOption = curOption->next;
   }

   /* Return x coordinate */
   *x = x1 + width + space;
}
```

```
/*****************************************************************
 * Function: displayPoint
 *
 * Parameters:
 *    im (gdImagePtr*): pointer to image data
 *    x (int*): x coordinate
 *    y (int): y coordinate
 *    node (char*): the name of the node
 *    list (char*): the name of the list
 *    rule (char*): the name of the rule
 *    option (char*): the desc of the option
 *    color (int): color number
 *    ftm (char*): string to display
 *
 * Returns: void
 *
 * Draw the point on the graph
 *****************************************************************/
void displayPoint (gdImagePtr *im, int x, int y, char *node,
      char *list, char *rule, char *option, int color, char *fmt, ...) {
   va_list ap;          /* String parameters */
   char str[1024],      /* Temp string */
      *tmp1, *tmp2, *tmp3;
   int ruleNum;         /* Decimal value of rule number */

   /* Create string */
   va_start (ap, fmt);
   vsprintf (str, fmt, ap);
   va_end (ap);

   /* Escape any single quotes */
   if (strchr (str, 39)) {
      tmp1 = (char*)malloc ((2 * strlen(str)) + 1);
      strcpy (tmp1, "");
      tmp2 = str;

      while ((tmp3 = strchr (tmp2, 39))) {
       memset (tmp3, '\0', 1);
       strcat (tmp1, tmp2);
       strcat (tmp1, "\\\"");
       tmp2 = tmp3 + 1;
      }
      strcat (tmp1, tmp2);
      strcpy (str, tmp1);
      free (tmp1);
   }

   /* Display point */
   gdImageFilledRectangle(*im, x, y, (x + width), (y + height), color);

   /* Display text in graph */
   if (strcmp (args.rule, "") != 0) {
      if (option) {
       if ((strcmp (args.rule, "") != 0) &&
            (strcmp (args.rule, "all") != 0) && (args.zoom == 0))
          gdImageString(*im, gdFontSmall, (x + width + space), y, option,
color);
```

```
      } else if (rule) {
       if ((strcmp (args.rule, "all") != 0) && (args.zoom == 0))
          gdImageString(*im, gdFontSmall, (x + width + space), y, rule,
color);
      } else if (list) {
       gdImageString(*im, gdFontSmall, (x + width + space), y, list, color);
      } else if (node) {
       gdImageString(*im, gdFontSmall, (x + width + space), y, node, color);
      }
   }

   /* Display html code */
   printf ("<AREA SHAPE=RECT COORDS=\"%d,%d,%d,%d\" HREF=\"",
                              x, y, (x + width), (y + height));

   /* Create path */
   if ((node) || (list) || (rule)) {
      printf ("%s/snortDisplay.pl?nodes=%s&lists=", BASECGI, node);

      /* Display the list selection */
      if (list)
       printf ("%s", list);
      else if (strcmp (node, "all") != 0)
       printf ("all");

      /* Display the rules selection */
      printf ("&rules=");
      if (rule) {
       ruleNum = atoi (strchr (rule, ' '));
       printf ("Rule%%20%d", ruleNum);
      } else if ((list) && (strcmp (list, "all") != 0)) {
       printf ("all");
      }

      /* Display the width, if provided */
      if (args.width)
       printf ("&width=%d", args.width);
   }

   /* Display event code */
   printf ("\" onMouseOver='showDetails (\"%s\", %d, %d)' "
                "onMouseOut='hideDetails ()'>\n", str, (x + 10), y);
}
```

```
/*****************************************************************
 * Function: displayLine
 *
 * Parameters:
 *    im (gdImagePtr*): pointer to image data
 *    head (void*): the head of the items
 *    next (void*): the next item
 *    x1 (int): start next x coordinate
 *    y1 (int): start next y coordinate
 *    x2 (int): end next x coordinate
 *    y2 (int): end next y coordinate
 *    x3 (int): start head x coordinate
 *    y3 (int): start head y coordinate
 *    x4 (int): end head x coordinate
 *    y4 (int): end head y coordinate
 *    color (int): color number
 *
 * Returns: void
 *
 * Draw the line(s)
 *****************************************************************/
void displayLine (gdImagePtr *im, void *head, void *next, int x1, int y1,
            int x2, int y2, int x3, int y3, int x4, int y4, int color) {
   /* Display a line to the next entry in the list */
   if (next)
      gdImageLine(*im, x1, y1, x2, y2, color);

   /* Display a line to the next level of the tree */
   if (head)
      gdImageLine(*im, x3, y3, x4, y4, color);
}

/*****************************************************************
 * Function: freeImage
 *
 * Parameters:
 *    root (node*): the root node
 *
 * Returns: void
 *
 * Free all the data structures
 *****************************************************************/
void freeImage (node *root) {
   node *curNode, *nextNode;
   list *curList, *nextList;
   rule *curRule, *nextRule;
   option *curOption, *nextOption;

   /* Start with the first node */
   curNode = root;

   /* Loop through all nodes */
   while (curNode != NULL) {
      /* Get the list */
      curList = curNode->head;

      /* Loop through all lists */
```

```
      while (curList != NULL) {
       /* Get the rule */
       curRule = curList->head;

       /* Loop through all the rules */
       while (curRule != NULL) {
          /* Get the option */
          curOption = curRule->head;

          /* Loop through all the options */
          while (curOption != NULL) {
            nextOption = curOption->next;
            free (curOption->name);
            free (curOption->msg);
            free (curOption);
            curOption = nextOption;
          }
          nextRule = curRule->next;
          free (curRule->name);
          free (curRule->src_port);
          free (curRule->dst_port);
          free (curRule->src_ip);
          free (curRule->dst_ip);
          free (curRule);
          curRule = nextRule;
       }
       nextList = curList->next;
       free (curList->name);
       free (curList);
       curList = nextList;
      }
      nextNode = curNode->next;
      free (curNode->name);
      free (curNode);
      curNode = nextNode;
   }
}

/*****************************************************************
 * Function: createIPString
 *
 * Parameters:
 *    str (char*): the original ip string
 *
 * Returns: (char*)
 *    the string with ":" replaced by "<BR>\n"
 *    NULL if an error occurs
 *
 * Create a new ip string
 *****************************************************************/
char *createIPString (char *str) {
   int len;       /* Length of original string */
   char *cpystr,  /* Copy of original string */
      *newstr,    /* The new string */
      *ptr;       /* Pointer to location within str */

   /* Initialize variables */
```

```
   cpystr = strdup (str);
   len = strlen (cpystr);
   newstr = (char*)malloc (len + ((len / 15) * 8) + 1);

   /* Create the new string */
   ptr = cpystr;
   strcpy (newstr, "");
   while (ptr != NULL) {
      if (strchr (ptr, ':')) {
       memset (strchr (ptr, ':'), '\0', 1);
             strcat (newstr, ptr);
       strcat (newstr, "<BR>\\n\\t");
       ptr += strlen (ptr) + 1;

      } else {
             strcat (newstr, ptr);
       strcat (newstr, "<BR>\\n");
       ptr = NULL;
      }
   }

   /* Free memory */
   free (cpystr);

   /* Done */
   return (newstr);
}
```

A.7. displayFunctions.h

```
/*************************************************************
 * Copyright 2003 Joseph Belans <belans@cs.fsu.edu>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *************************************************************/
#include <gd.h>

#ifndef _DISPLAYFUNCTIONS

#define _DISPLAYFUNCTIONS

#define FRONT 1
```

```
#define MIDDLE 2
#define END 3

/* Declare structures */
/* Option structure */
typedef struct _option {
   char *name;
   char *msg;
   int id;
   int rev;
   struct _option *next;
}option;

/* Rule structure */
typedef struct _rule {
   char *name;
   char *src_port;
   char *dst_port;
   char *src_ip;
   char *dst_ip;
   int numOptions;
   struct _rule *prev;
   struct _rule *next;
   option *head;
}rule;

/* List structure */
typedef struct _list {
   char *name;
   int numRules;
   int numOptions;
   int maxOptions;
   struct _list *next;
   rule *head;
}list;

/* Node structure */
typedef struct _node {
   char *name;
   int numLists;
   int numRules;
   int numOptions;
   int maxOptions;
   struct _node *next;
   list *head;
}node;

/* Command line arguments */
struct _args {
   char *node;
   char *list;
   char *rule;
   int location;
   int zoom;
   int width;
}args;
```

```
/* Declare functions */
void drawImage (node *root, int maxOptions, char *file);
void drawNodes (gdImagePtr *im, node *head, int x, int y);
void drawLists (gdImagePtr *im, list *head, char *node, int *x, int y);
void drawRules (gdImagePtr *im, rule *head,
                  char *node, char *list, int *x, int y);
void drawOptions (gdImagePtr *im, option *head, int *x, int y);
void freeImage (node *root);
void displayArea (int x1, int y1, int x2, int y2, char *fmt, ...);

#endif
```

A.8. header.pl

```
############################################################
# Copyright 2003 Joseph Belans <belans@cs.fsu.edu>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
############################################################
############################################################
# File: header.pl
#
# Parameters: (read in from stdin)
#       file: path to configuration file
#
# Returns: none
#
# Receives file name from stdin via cgi.  Will process the
# path or file selected and display the appropriate html
############################################################
# Require all variables are declared prior to use
use strict;

<INSERT>

############################################################
# Function: validPath
#
# Parameters:
#       file: path to configuration file
#
# Returns: none
```

```
#
# Make sure BASECONFDIR is the base directory
############################################################
sub validPath {
   my ($file) = shift;
   my ($str) = BASECONFDIR;

   $str =~ s/\//\\\//g;
   $str = "\$file !~ /^$str/";

   # Invalid path
   if (eval ($str)) {
      return (0);
   }

   # Valid path
   return (1);
}

return 1;
```

A.9. snortDisplay.html.in

```
<HTML>
<HEAD>
<!--
Copyright 2003 Joseph Belans <belans@cs.fsu.edu>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
-->
<SCRIPT LANGUAGE="JavaScript">
// Check for the enter key
var Nav4 = document.layers;
var IE4 = document.all;

//************************************************
// Function: onEnter
//
// Parameters:
//    e: the event
//
// Returns: none
//
```

```
// Submit form data when enter key is pressed
//************************************************
function onEnter(e) {
   if (Nav4)
      keyPressed = String.fromCharCode(e.which);
   else if (IE4)
      keyPressed = String.fromCharCode(window.event.keyCode);

   if (keyPressed.charCodeAt(0) == 13) {
      document.display.submit();
   }
}

//************************************************
// Function: openBrowseWin
//
// Parameters: none
//
// Returns: none
//
// Open a new browser window to browse for config file
//************************************************
function openBrowseWin () {
   window.open('<VIRTUAL_CGI_PATH>/browse.pl', 'Browse',
      'toolbar=0, status=0, resizable=0, width=250, height=500');
}

//************************************************
// Function: openBrowseWin
//
// Parameters: none
//
// Returns: none
//
// Open a new browser window to browse for config file
//************************************************
function menuSelect (select) {
   // Set the remaining lists to the 0 index
   if (select.name == "nodes") {
      document.display.lists.selectedIndex = 0;
      document.display.rules.selectedIndex = 0;

   } else if (select.name == "lists") {
      document.display.rules.selectedIndex = 0;
   }
}

//************************************************
// Function: showDetails
//
// Parameters:
//    str: the name of the point
//    x: x coordinate
//    y: y coordinate
//
// Returns: none
//
```

```
// Displays pop-up window containing information about point
//************************************************
function showDetails (str, x, y) {
   if (document.getElementById) {
      document.getElementById("details").innerHTML = str;
      document.getElementById("details").style.visibility = "visible";
      document.getElementById("details").style.left = x + 10;
      document.getElementById("details").style.top = y;
   } else {
      document.details.document.open();
      document.details.document.write(str);
      document.details.document.close();
      document.details.visibility = "visible";
      document.details.left = x + 10;
      document.details.top = y;
   }
}

//************************************************
// Function: hideDetails
//
// Parameters: none
//
// Returns: none
//
// Hide the pop-up window
//************************************************
function hideDetails () {
   if (document.getElementById) {
      document.getElementById("details").style.visibility = "hidden";
   } else {
      document.details.visibility = "hidden";
   }
}
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT>
// Capture events
if (window.document.captureEvents!=null)
   window.document.captureEvents(Event.KEYPRESS)
window.document.onkeypress = onEnter;
</SCRIPT>
<TABLE WIDTH="100%"><TR><TD>
<P>
<FORM NAME="config_form" METHOD="post"
ACTION="<VIRTUAL_CGI_PATH>/snortDisplay.pl">
Config file: <INPUT TYPE="text" NAME="config"><!--config--></INPUT>
<INPUT TYPE="submit" NAME="submit" VALUE="Load"></INPUT>
<INPUT TYPE="submit" NAME="submit" VALUE="Browse"
      onClick="openBrowseWin (); return (false);"></INPUT>
</FORM>
</TD><TD ALIGN="center" VALIGN="center">
<H1>SNORT DISPLAY</H1>
</TD><TD ALIGN="right">
<FORM NAME="display" METHOD="post"
ACTION="<VIRTUAL_CGI_PATH>/snortDisplay.pl">
```

```
<INPUT TYPE=hidden NAME=changed VALUE="">
<INPUT TYPE=hidden NAME=width>
<TABLE>
<TR><TH>Node</TH><TH>Protocol</TH><TH>Rule</TH></TR>
<TR>
<TD ALIGN="center">
<SELECT NAME=nodes onChange="this.form.submit()"><!--nodes--></SELECT>
</TD>
<TD ALIGN="center">
<SELECT NAME=lists onChange="this.form.submit()"><!--lists--></SELECT>
</TD>
<TD ALIGN="center">
<SELECT NAME=rules onChange="this.form.submit()"><!--rules--></SELECT>
</TD>
</TR><TR>
<TD COLSPAN=3>
<!--zoom-->
</TD>
</TR></TABLE>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
// Set the width of the screen
document.display.width.value = screen.availWidth;
</SCRIPT>
</TD></TR></TABLE>
<P>
<DIV id="details" style="position: absolute; z-index: 1; top: 200px; left:
100px; width: 300px; border: solid 3px #000000; background-color: #ffff00;
visibility: hidden;"></DIV>
<P>
<MAP NAME="mainmap">
<!--display-->
</BODY>
</HTML>
```

A.10. snortDisplay.pl.in

```
#!/usr/bin/perl

##############################################################
# Copyright 2003 Joseph Belans <belans@cs.fsu.edu>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
```

```
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#############################################################
#############################################################
# File: snortDisplay.pl
#
# Parameters: (read in from stdin)
#       file: path to configuration file
#
# Returns: none
#
# Receives file name from stdin via cgi.  Will process the
# path or file selected and display the appropriate html
#############################################################
# Require all variables are declared prior to use
use strict;

# Include cgi functions and header information
require "process_cgi.pl";
BEGIN { require "header.pl" }

# If the template file does not exist, display error
if ( ! -f TEMPLATE ) {
   print_header ();
   print "Template does not exist: " . TEMPLATE;
   exit;
}

# Read in the template
open (INFILE, TEMPLATE);
my (@MENU) = <INFILE>;
close (INFILE);

# Get the form data
my (%FORM);
parse_input (\%FORM);

# Load the data file
if ($FORM{'submit'} eq "Load") {
   system (BASECGIDIR . "/config $FORM{'config'} " . DATAFILE);
   print_header ($ENV{"REQUEST_URI"} . "\n\n");
   exit;

# If the data file does not exist, display empty template
} elsif ( ! -f DATAFILE ) {
   print_header ();
   print @MENU;
   exit;

# Display the header
} else {
   print_header ();
}

# Read in the data file
open (INFILE, DATAFILE);
my (@DATA) = <INFILE>;
close (INFILE);
```

```
# Strip the newline from all entries
foreach (@DATA) {
   chop;
}

# Reset form data based on selections
if ($FORM{'nodes'} eq "all") {
   $FORM{'lists'} = "";
   $FORM{'rules'} = "";
   $FORM{'zoom'} = 0;
} elsif ($FORM{'lists'} eq "all") {
   $FORM{'rules'} = "";
   $FORM{'zoom'} = 0;
} elsif ($FORM{'rules'} eq "all") {
   $FORM{'zoom'} = 0;
}

# Get the config file name
my $config = $DATA[0];
shift (@DATA);

# Insert the data
my (@tmp, @desc, @value, $tmp, $item, $i);
foreach $item (@MENU) {
   # Insert the config file name
   if ($item =~ /<!--config-->/) {
      eval ("\$item =~ s|><!--config--| VALUE = \"" . $config . "\"|");
      print $item;

   # Create the graph
   } elsif (($item =~ /<!--display-->/) && (($FORM{'nodes'} eq "all") ||
                  ($FORM{'lists'} eq "all") || ($FORM{'rules'} ne ""))) {
      my $cmd = BASECGIDIR . "/display \"$FORM{'nodes'}\" " .
            "\"$FORM{'lists'}\" \"$FORM{'rules'}\" \"$FORM{'zoom'}\" " .
            "\"$FORM{'width'}\" \"$FORM{'location'}\"";
      print `$cmd`;

   # Fill in the node menu
   } elsif ($item =~ /<!--nodes-->/) {
      @value = grep (/^\w/, @DATA);
      foreach (@value) {
       push (@desc, ucfirst ($_));
      }
      fillSelect ($item, "nodes", $FORM{'nodes'}, \@value, \@desc);
      @value = ();
      @desc = ();

   # Continue if a node was selected
   } elsif (($FORM{'nodes'} ne "")  && ($FORM{'nodes'} ne "all")) {
      if ($item =~ /<!--lists-->/) {
       for ($i = 0; $DATA[$i]; $i++) {
          if (eval ("\$DATA\[\$i\] =~ /^" . $FORM{'nodes'} . "/")) {
            for ($i++; $DATA[$i]; $i++) {
               if ($DATA[$i] =~ /^ (\w.*)$/) {
                  push (@value, $1);
                  push (@desc, $1);
```

```
            } elsif ($DATA[$i] =~ /^\w/) {
               last;
            }
          }

          fillSelect ($item, "lists", $FORM{'lists'}, \@value, \@desc);
          @value = ();
          @desc = ();
          last;
        }
       }

      # Continue if a list was selected
      } elsif (($FORM{'lists'} ne "") && ($FORM{'lists'} ne "all")) {
       # Fill in the rule menu
       if ($item =~ /<!--rules-->/) {
          for ($i = 0; $DATA[$i]; $i++) {
            if (eval ("\$DATA\[\$i\] =~ /^" . $FORM{'nodes'} . "/")) {
               for ($i++; $DATA[$i]; $i++) {
                  if (eval ("\$DATA\[\$i\] =~ /^ " . $FORM{'lists'} . "/")) {
                   for ($i++; $DATA[$i]; $i++) {
                      if ($DATA[$i] =~ /^  (Rule \d+):(.*)$/) {
                        push (@value, "$1");
                        push (@desc, $1);

                      } elsif ($DATA[$i] =~ /^ \w/) {
                        last;
                      }
                   }
                  }
               }
               fillSelect ($item, "rules", $FORM{'rules'}, \@value, \@desc);
               @value = ();
               @desc = ();
               last;
            }
          }

       # Display the zoom fields
       } elsif ($item =~ /<!--zoom-->/) {
          print "<B>Number of rules to show:</B>\n";

          if ($FORM{'zoom'} gt 0) {
            print "<INPUT TYPE=text NAME=zoom VALUE=$FORM{'zoom'} SIZE=3
MAXLENGTH=3><BR>\n";
          } else {
            print "<INPUT TYPE=text NAME=zoom VALUE=0 SIZE=3
MAXLENGTH=3><BR>\n";
          }

          print "<B>Selection:\n";

          print "Front <INPUT TYPE=radio NAME=location VALUE=front
onChange=\"this.form.submit()\"";
          if (($FORM{'location'} eq "") || ($FORM{'location'} eq "front")) {
            print " CHECKED";
```

```
          }
          print ">\n";

          print "Middle <INPUT TYPE=radio NAME=location VALUE=middle
onChange=\"this.form.submit()\"";
          if ($FORM{'location'} eq "middle") {
            print " CHECKED";
          }
          print ">\n";

          print "End <INPUT TYPE=radio NAME=location VALUE=end
onChange=\"this.form.submit()\"";
          if ($FORM{'location'} eq "end") {
            print " CHECKED";
          }
          print ">\n";

       } else {
          print $item;
       }

      } else {
       print $item;
      }

   # Otherwise, print the line
   } else {
      print $item;
   }
}
exit;

#########################################################################
# Function: fillSelect
#
# Parameters:
#     line - next line from the data file
#     str - the name of the drop down box
#     select - item of drop down box that is selected
#     value - address of array containing values
#     desc - address of array containing descriptions
#
# Return: none
#
# Fills in the select box
#########################################################################
sub fillSelect {
   my ($line) = shift;
   my ($str) = shift;
   my ($select) = shift;
   my ($value) = shift;
   my ($desc) = shift;
   my ($i, $tmp);

   # Generate the substitution string
   # Insert the all selection
   $tmp = "\n<OPTION>\n<OPTION VALUE=\"all\"";
```

```
   if ($select eq "all") {
      $tmp .= " SELECTED";
   }
   $tmp .= ">All\n";

   # Insert the data values
   for ($i = 0; $$value[$i]; $i++) {
      $tmp .= "<OPTION VALUE=\"$$value[$i]\"";

      if ($$value[$i] eq $select) {
       $tmp .= " SELECTED";
      }

      $tmp .= ">$$desc[$i]\n";
   }

   eval ("\$line =~ s|<!--" . $str . "-->|" . $tmp . "|");
   print "$line";
}
```

APPENDIX B

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it

if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and
modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to

this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally

print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals

of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY

YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) year name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
> This is free software, and you are welcome to redistribute it
> under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989

Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

REFERENCES

1. M. Roesch, Snort Users Manual: Snort Release 1.9.x, 26 April 2002.

2. M. Roesch, Snort – lightweight intrusion detection for networks, Proceedings of LISA 99, 1999, http://www.snort.org/docs/lisa-paper.txt

3. D. Sequeira, Intrusion Prevention Systems – Security's Silver Bullet?, November 14, 2002, http://www.sans.org/rr/intrusion/silver_bullet.php

4. Top Layer Networks, Beyond IDS: Essentials of Network Intrusion Prevention, November 2002

5. T. Verwoerd, R. Hunt, Intrusion detection techniques and approaches, Computer Communications, Volume 25, Issue 15, 15 September 2002, Pages 1356-1365

6. M. Williams, Study: Slammer was fastest spreading worm yet, InfoWorld, 03 February 2003, http://www.infoworld.com/article/03/02/03/HNslamfast_1.html?applications

7. Snort, http://www.snort.org/

8. ACID, http://www.andrew.cmu.edu/~rdanyliw/snort/snortacid.html

9. Webmin, http://www.webmin.com/

10. Cisco, http://www.cisco.com/

11. Enterasys, http://www.enterasys.com/home.html

BIOGRAPHICAL SKETCH

Shortly after graduating with a Bachelors of Science in Computer Science in April of 1997 from The Florida State University (FSU), I began employment as a programmer for a unit within the university, the Florida Information Resource Network (FIRN). Within the first couple of years with FIRN, I not only progressed very quickly within the organization, but my knowledge was progressing even faster. I feel that entering the "real world" had a great effect on my success in the master's program at FSU. After entering the master's program, I gained a great interest in systems administration and decided to continue with the systems administration track. In time, I took over as a Solaris administrator for FSU. Again, after assuming the new role, my interests narrowed further into the realm of security. With that in mind, I elected to do my project on Snort because I wanted a project that had more of a "real life" feel in security rather than a theoretical project. Once I graduate with a master's degree, I would like to continue on with my interest in security and enter a position with the Federal Government in one of the many cyber security divisions.