CHAPTER 1

Introduction to Intrusion Detection and Snort

Security is a big issue for all networks in today's enterprise environment. Hackers and intruders have made many successful attempts to bring down high-profile company networks and web services. Many methods have been developed to secure the network infrastructure and communication over the Internet, among them the use of firewalls, encryption, and virtual private networks. Intrusion detection is a relatively new addition to such techniques. Intrusion detection methods started appearing in the last few years. Using intrusion detection methods, you can collect and use information from known types of attacks and find out if someone is trying to attack your network or particular hosts. The information collected this way can be used to harden your network security, as well as for legal purposes. Both commercial and open source products are now available for this purpose. Many vulnerability assessment tools are also available in the market that can be used to assess different types of security holes present in your network. A comprehensive security system consists of multiple tools, including:

- Firewalls that are used to block unwanted incoming as well as outgoing traffic of data. There is a range of firewall products available in the market both in Open Source and commercial products. Most popular commercial firewall products are from Checkpoint (http://www.checkpoint.com), Cisco (http://www.cisco.com) and Netscreen

(http://www.netscreen.com). The most popular Open Source firewall is the Netfilter/Iptables (http://www.netfilter.org)-based firewall.

- Intrusion detection systems (IDS) that are used to find out if someone has gotten into or is trying to get into your network. The most popular IDS is Snort, which is available at http://www.snort.org.
- Vulnerability assessment tools that are used to find and plug security holes present in your network. Information collected from vulnerability assessment tools is used to set rules on firewalls so that these security holes are safeguarded from malicious Internet users. There are many vulnerability assessment tools including Nmap (http://www.nmap.org) and Nessus (http://www.nessus.org).

These tools can work together and exchange information with each other. Some products provide complete systems consisting of all of these products bundled together.

Snort is an open source *Network Intrusion Detection System* (NIDS) which is available free of cost. NIDS is the type of Intrusion Detection System (IDS) that is used for scanning data flowing on the network. There are also host-based intrusion detection systems, which are installed on a particular host and detect attacks targeted to that host only. Although all intrusion detection methods are still new, Snort is ranked among the top quality systems available today.

The book starts with an introduction to intrusion detection and related terminology. You will learn installation and management of Snort as well as other products that work with Snort. These products include MySQL database (http://www.mysql.org) and Analysis Control for Intrusion Database (ACID) (http://www.cert.org/kb/acid). Snort has the capability to log data collected (such as alerts and other log messages) to a database. MySQL is used as the database engine where all of this data is stored. Using Apache web server (http://www.apache.org) and ACID, you can analyze this data. A combination of Snort, Apache, MySQL, and ACID makes it possible to log the intrusion detection data into a database and then view and analyze it later, using a web interface.

This book is organized in such a way that the reader will be able to build a complete intrusion detection system by going through the following chapters in a step-by-step manner. All steps of installing and integrating different tools are explained in the book as outlined below.

Chapter 2 provides basic information about how to build and install Snort itself. Using the basic installation and default rules, you will be able to get a working IDS. You will be able to create log files that show intrusion activity.

Chapter 3 provides information about Snort rules, different parts of Snort rules and how to write your own rules according to your environment and needs. This chapter

is very important, as writing good rules is the key to building a detection system. The chapter also explains different rules that are part of Snort distribution.

Chapter 4 is about input and output plug-ins. Plug-ins are parts of the software that are compiled with Snort and are used to modify input or output of the Snort detection engine. Input plug-ins prepare captured data packets before the actual detection process is applied on these packets. Output plug-ins format output to be used for a particular purpose. For example, an output plug-in can convert the detection data to a Simple Network Management Protocol (SNMP) trap. Another output plug-in is used to log Snort output data into databases. This chapter provides a comprehensive overview of how these plug-ins are configured and used.

Chapter 5 provides information about using MySQL database with Snort. MySQL plug-in enables Snort to log data into the database to be used in the analysis later on. In this chapter you will find information about how to create a database in MySQL, configure a database plug-in, and log data to the database.

Chapter 6 describes ACID, how to use it to get data from the database you configured in Chapter 5, and how to display it using Apache web server. ACID is a very important tool that provides rich data analysis capabilities. You can find frequency of attacks, classify different attacks, view the source of these attacks and so on. ACID uses PHP (Pretty Home Page) scripting language, graphic display library (GD library) and PHPLOT, which is a tool to draw graphs. A combination of all of these results in web pages that display, analyze and graph data stored in the MySQL database.

Chapter 7 is devoted to information about some other useful tools that can be used with Snort.

The system that you will build after going through this book is displayed in Figure 1-1 with different components.

As you can see, data is captured and analyzed by Snort. Snort then stores this data in the MySQL database using the database output plug-in. Apache web server takes help from ACID, PHP, GD library and PHPLOT package to display this data in a browser window when a user connects to Apache. A user can then make different types of queries on the forms displayed in the web pages to analyze, archive, graph and delete data.

In essence, you can build a single computer with Snort, MySQL database, Apache, PHP, ACID, GD library and PHPLOT. A more realistic picture of the system that you will be able to build after reading this book is shown in Figure 1-2.

In the enterprise, usually people have multiple Snort sensors behind every router or firewall. In that case you can use a single centralized database to collect data from all of the sensors. You can run Apache web server on this centralized database server as shown in Figure 1-3.

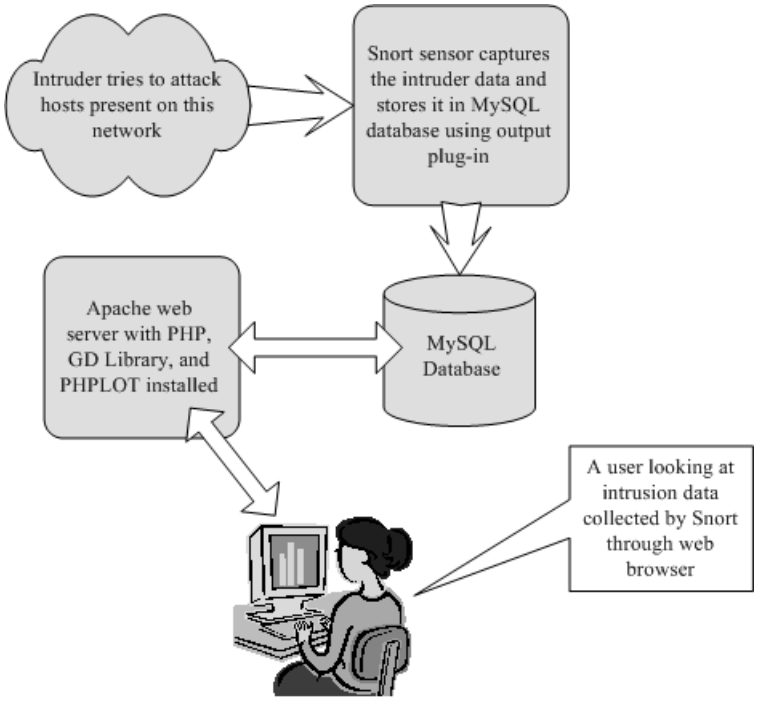

Figure 1-1 Block diagram of a complete network intrusion detection system consisting of Snort, MySQL, Apache, ACID, PHP, GD Library and PHPLOT.

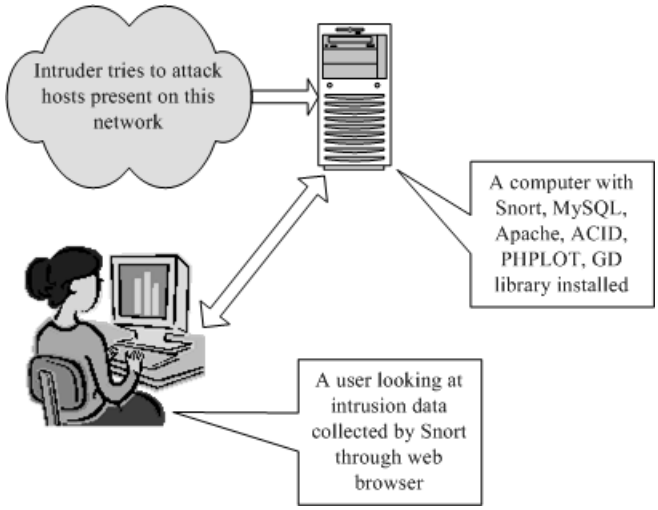

Figure 1-2 A network intrusion detection system with web interface.

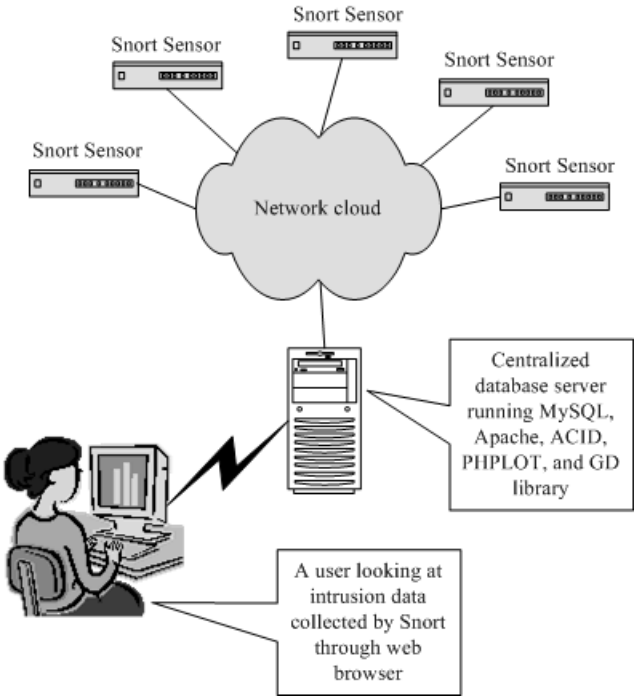

Figure 1-3 Multiple Snort sensors in the enterprise logging to a centralized database server.

1.1 What is Intrusion Detection?

Intrusion detection is a set of techniques and methods that are used to detect suspicious activity both at the network and host level. Intrusion detection systems fall into two basic categories: signature-based intrusion detection systems and anomaly detection systems. Intruders have signatures, like computer viruses, that can be detected using software. You try to find data packets that contain any known intrusion-related signatures or anomalies related to Internet protocols. Based upon a set of signatures and rules, the detection system is able to find and log suspicious activity and generate alerts. Anomaly-based intrusion detection usually depends on packet anomalies present in protocol header parts. In some cases these methods produce better results compared to signature-based IDS. Usually an intrusion detection system captures data from the network and applies its rules to that data or detects anomalies in it. Snort is primarily a rule-based IDS, however input plug-ins are present to detect anomalies in protocol headers.

Snort uses rules stored in text files that can be modified by a text editor. Rules are grouped in categories. Rules belonging to each category are stored in separate files. These files are then included in a main configuration file called snort.conf. Snort reads these rules at the start-up time and builds internal data structures or chains to apply these rules to captured data. Finding signatures and using them in rules is a tricky job, since the more rules you use, the more processing power is required to process captured data in real time. It is important to implement as many signatures as you can using as few rules as possible. Snort comes with a rich set of pre-defined rules to detect intrusion activity and you are free to add your own rules at will. You can also remove some of the built-in rules to avoid false alarms.

1.1.1 Some Definitions

Before we go into details of intrusion detection and Snort, you need to learn some definitions related to security. These definitions will be used in this book repeatedly in the coming chapters. A basic understanding of these terms is necessary to digest other complicated security concepts.

1.1.1.1 IDS

Intrusion Detection System or IDS is software, hardware or combination of both used to detect intruder activity. Snort is an open source IDS available to the general public. An IDS may have different capabilities depending upon how complex and sophisticated the components are. IDS appliances that are a combination of hardware and software are available from many companies. As mentioned earlier, an IDS may use signatures, anomaly-based techniques or both.

1.1.1.2 Network IDS or NIDS

NIDS are intrusion detection systems that capture data packets traveling on the network media (cables, wireless) and match them to a database of signatures. Depending upon whether a packet is matched with an intruder signature, an alert is generated or the packet is logged to a file or database. One major use of Snort is as a NIDS.

1.1.1.3 Host IDS or HIDS

Host-based intrusion detection systems or HIDS are installed as agents on a host. These intrusion detection systems can look into system and application log files to detect any intruder activity. Some of these systems are reactive, meaning that they inform you only when something has happened. Some HIDS are proactive; they can sniff the network traffic coming to a particular host on which the HIDS is installed and alert you in real time.

1.1.1.4 Signatures

Signature is the pattern that you look for inside a data packet. A signature is used to detect one or multiple types of attacks. For example, the presence of "scripts/iisadmin" in a packet going to your web server may indicate an intruder activity.

Signatures may be present in different parts of a data packet depending upon the nature of the attack. For example, you can find signatures in the IP header, transport layer header (TCP or UDP header) and/or application layer header or payload. You will learn more about signatures later in this book.

Usually IDS depends upon signatures to find out about intruder activity. Some vendor-specific IDS need updates from the vendor to add new signatures when a new type of attack is discovered. In other IDS, like Snort, you can update signatures yourself.

1.1.1.5 Alerts

Alerts are any sort of user notification of an intruder activity. When an IDS detects an intruder, it has to inform security administrator about this using alerts. Alerts may be in the form of pop-up windows, logging to a console, sending e-mail and so on. Alerts are also stored in log files or databases where they can be viewed later on by security experts. You will find detailed information about alerts later in this book.

Snort can generate alerts in many forms and are controlled by output plug-ins. Snort can also send the same alert to multiple destinations. For example, it is possible to log alerts into a database and generate SNMP traps simultaneously. Some plug-ins can also modify firewall configuration so that offending hosts are blocked at the firewall or router level.

1.1.1.6 Logs

The log messages are usually saved in file. By default Snort saves these messages under /var/log/snort directory. However, the location of log messages can be changed using the command line switch when starting Snort. Log messages can be saved either in text or binary format. The binary files can be viewed later on using Snort or tcpdump program. A new tool called Barnyard is also available now to analyze binary log files generated by Snort. Logging in binary format is faster because it saves some formatting overhead. In high-speed Snort implementations, logging in binary mode is necessary.

1.1.1.7 False Alarms

False alarms are alerts generated due to an indication that is not an intruder activity. For example, misconfigured internal hosts may sometimes broadcast messages that trigger a rule resulting in generation of a false alert. Some routers, like Linksys home routers, generate lots of UPnP related alerts. To avoid false alarms, you have to modify

and tune different default rules. In some cases you may need to disable some of the rules to avoid false alarms.

1.1.1.8 Sensor

The machine on which an intrusion detection system is running is also called the sensor in the literature because it is used to "sense" the network. Later in this book if the word sensor is used, it refers to a computer or other device where Snort is running.

1.1.2 Where IDS Should be Placed in Network Topology

Depending upon your network topology, you may want to position intrusion detection systems at one or more places. It also depends upon what type of intrusion activities you want to detect: internal, external or both. For example, if you want to detect only external intrusion activities, and you have only one router connecting to the Internet, the best place for an intrusion detection system may be just inside the router or a firewall. If you have multiple paths to the Internet, you may want to place one IDS box at every entry point. However if you want to detect internal threats as well, you may want to place a box in every network segment.

In many cases you don't need to have intrusion detection activity in all network segments and you may want to limit it only to sensitive network areas. Note that more intrusion detection systems mean more work and more maintenance costs. Your decision really depends upon your security policy, which defines what you really want to protect from hackers. Figure 1-4 shows typical locations where you can place an intrusion detection system.

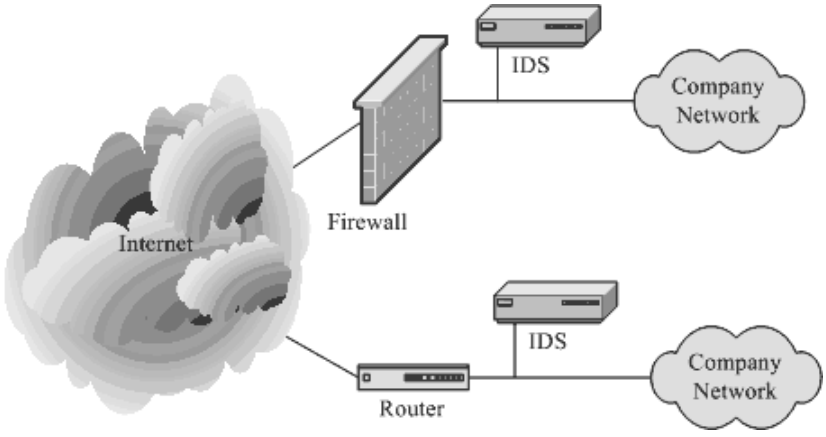

Figure 1-4 Typical locations for an intrusion detection system.

As you can see from Figure 1-4, typically you should place an IDS behind each of your firewalls and routers. In case your network contains a demilitarized zone (DMZ), an IDS may be placed in that zone as well. However alert generation policy should not be as strict in a DMZ compared to private parts of the network.

1.1.3 Honey Pots

Honey pots are systems used to lure hackers by exposing known vulnerabilities deliberately. Once a hacker finds a honey pot, it is more likely that the hacker will stick around for some time. During this time you can log hacker activities to find out his/her actions and techniques. Once you know these techniques, you can use this information later on to harden security on your actual servers.

There are different ways to build and place honey pots. The honey pot should have common services running on it. These common services include Telnet server (port 23), Hyper Text Transfer Protocol (HTTP) server (port 80), File Transfer Protocol (FTP) server (port 21) and so on. You should place the honey pot somewhere close to your production server so that the hackers can easily take it for a real server. For example, if your production servers have Internet Protocol (IP) addresses 192.168.10.21 and 192.168.10.23, you can assign an IP address of 192.168.10.22 to the honey pot. You can also configure your firewall and/or router to redirect traffic on some ports to a honey pot where the intruder thinks that he/she is connecting to a real server. You should be careful in creating an alert mechanism so that when your honey pot is compromised, you are notified immediately. It is a good idea to keep log files on some other machine so that when the honey pot is compromised, the hacker does not have the ability to delete these files.

So when should you install a honey pot? The answer depends on different criteria, including the following:

- You should create a honey pot if your organization has enough resources to track down hackers. These resources include both hardware and personnel. If you don't have these resources, there is no need to install a honey pot. After all, there is no need to have data if you can't use it.
- A honey pot is useful only if you want to use the information gathered in some way.
- You may also use a honey pot if you want to prosecute hackers by gathering evidence of their activities.

Ideally a honey pot should look like a real system. You should create some fake data files, user accounts and so on to ensure a hacker that this is a real system. This will tempt the hacker to remain on the honey pot for a longer time and you will be able to record more activity.

To have more information and get a closer look at honey pots, go to the Honey Pot Project web site http://project.honeynet.org/ where you will find interesting material. Also go to the Honeyd web site at http://www.citi.umich.edu/u/provos/honeyd/ to find out information about this open source honey pot. Some other places where you can find more information are:

- South Florida Honeynet Project at http://www.sfhn.net
- Different HOWTOs at http://www.sfhn.net/whites/howtos.html

1.1.4 Security Zones and Levels of Trust

Some time ago people divided networks into two broad areas, secure area and unsecure area. Sometimes this division also meant a network is inside a firewall or a router and outside your router. Now typical networks are divided into many different areas and each area may have a different level of security policy and level of trust. For example, a company's finance department may have a very high security level and may allow only a few services to operate in that area. No Internet service may be available from the finance department. However a DMZ or de-militarized zone part of your network may be open to the Internet world and may have a very different level of trust.

Depending upon the level of trust and your security policy, you should also have different policies and rules for intruder detection in different areas of your network. Network segments with different security requirements and trust levels are kept physically separate from each other. You can install one intrusion detection system in each zone with different types of rules to detect suspicious network activity. As an example, if your finance department has no web server, any traffic going to port 80 in the finance department segment may come under scrutiny for intruder activity. The same is not true in the DMZ zone where you are running a company web server accessible to everyone.

1.2 IDS Policy

Before you install the intrusion detection system on your network, you must have a policy to detect intruders and take action when you find such activity. A policy must dictate IDS rules and how they will be applied. The IDS policy should contain the following components; you can add more depending upon your requirements.

- Who will monitor the IDS? Depending on the IDS, you may have alerting mechanisms that provide information about intruder activity. These alerting systems may be in the form of simple text files, or they may be more complicated, perhaps integrated to centralized network management systems like HP OpenView or MySQL database. Someone is needed to monitor the intruder activity and the policy must define the responsible person(s). The intruder activity may also be monitored in real time using pop-up windows or web interfaces. In this case operators must have knowledge of alerts and their meaning in terms of severity levels.
- Who will administer the IDS, rotate logs and so on? As with all systems, you need to establish routine maintenance of the IDS.
- Who will handle incidents and how? If there is no incident handling, there is no point in installing an IDS. Depending upon the severity of the incident, you may need to get some government agencies involved.
- What will be the escalation process (level 1, level 2 and so on)? The escalation process is basically an incident response strategy. The policy should clearly describe which incidents should be escalated to higher management.
- Reporting. Reports may be generated showing what happened during the last day, week or month.
- Signature updates. Hackers are continuously creating new types of attacks. These attacks are detected by the IDS if it knows about the attack in the form of signatures. Attack signatures are used in Snort rules to detect attacks. Because of the continuously changing nature of attacks, you must update signatures and rules on your IDS. You can update signatures directly from the Snort web site on a periodic basis or on your own when a new threat is discovered.
- Documentation is required for every project. The IDS policy should describe what type of documentation will be done when attacks are detected. The documentation may include a simple log or record of complete intruder activity. You may also need to build some forms to record data. Reports are also part of regular documentation.

Based on the IDS policy you will get a clear idea of how many IDS sensors and other resources are required for your network. With this information, you will be able to calculate the cost of ownership of IDS more precisely.

1.3 Components of Snort

Snort is logically divided into multiple components. These components work together to detect particular attacks and to generate output in a required format from the detection system. A Snort-based IDS consists of the following major components:

- Packet Decoder
- Preprocessors
- Detection Engine
- Logging and Alerting System
- Output Modules

Figure 1-5 shows how these components are arranged. Any data packet coming from the Internet enters the packet decoder. On its way towards the output modules, it is either dropped, logged or an alert is generated.

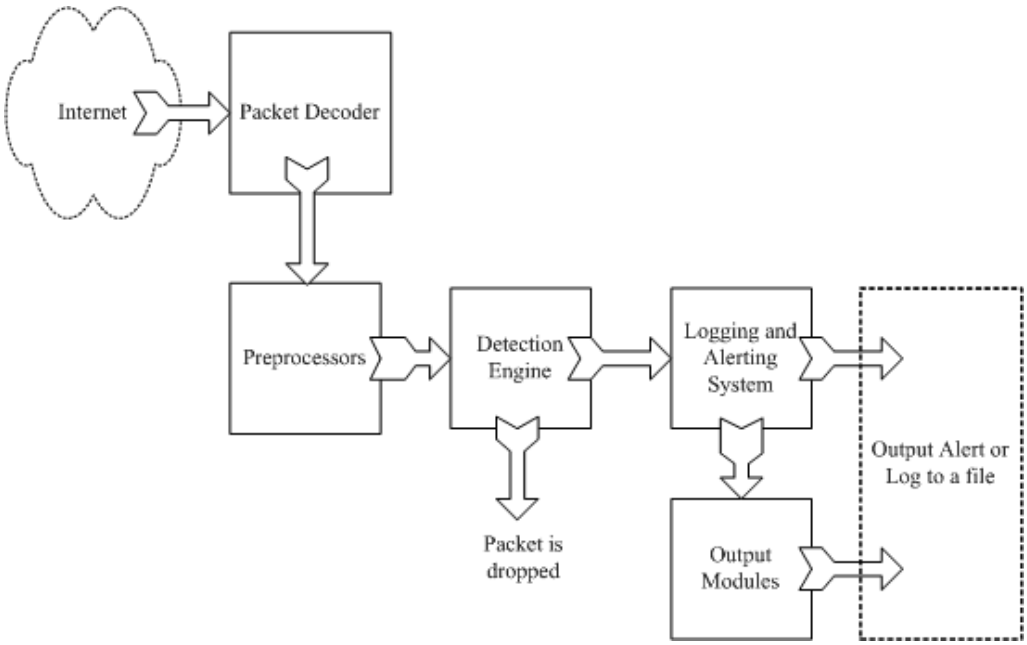

Figure 1-5 Components of Snort.

A brief introduction to these components is presented in this section. As you go through the book and create some rules, you will become more familiar with these components and how they interact with each other.

1.3.1 Packet Decoder

The packet decoder takes packets from different types of network interfaces and prepares the packets to be preprocessed or to be sent to the detection engine. The interfaces may be Ethernet, SLIP, PPP and so on.

1.3.2 Preprocessors

Preprocessors are components or plug-ins that can be used with Snort to arrange or modify data packets before the detection engine does some operation to find out if the packet is being used by an intruder. Some preprocessors also perform detection by finding anomalies in packet headers and generating alerts. Preprocessors are very important for any IDS to prepare data packets to be analyzed against rules in the detection engine. Hackers use different techniques to fool an IDS in different ways. For example, you may have created a rule to find a signature "scripts/iisadmin" in HTTP packets. If you are matching this string exactly, you can easily be fooled by a hacker who makes slight modifications to this string. For example:

- "scripts/./iisadmin"
- "scripts/examples/../iisadmin"
- "scripts\iisadmin"
- "scripts/.\iisadmin"

To complicate the situation, hackers can also insert in the web Uniform Resource Identifier (URI) hexadecimal characters or Unicode characters which are perfectly legal as far as the web server is concerned. Note that the web servers usually understand all of these strings and are able to preprocess them to extract the intended string "scripts/iisadmin". However if the IDS is looking for an exact match, it is not able to detect this attack. A preprocessor can rearrange the string so that it is detectable by the IDS.

Preprocessors are also used for packet defragmentation. When a large data chunk is transferred to a host, the packet is usually fragmented. For example, default maximum length of any data packet on an Ethernet network is usually 1500 bytes. This value is controlled by the Maximum Transfer Unit (MTU) value for the network interface. This means that if you send data which is more than 1500 bytes, it will be split into multiple data packets so that each packet fragment is less than or equal to 1500 bytes. The

receiving systems are capable of reassembling these smaller units again to form the original data packet. On IDS, before you can apply any rules or try to find a signature, you have to reassemble the packet. For example, half of the signature may be present in one segment and the other half in another segment. To detect the signature correctly you have to combine all packet segments. Hackers use fragmentation to defeat intrusion detection systems.

The preprocessors are used to safeguard against these attacks. Preprocessors in Snort can defragment packets, decode HTTP URI, re-assemble TCP streams and so on. These functions are a very important part of the intrusion detection system.

1.3.3 The Detection Engine

The detection engine is the most important part of Snort. Its responsibility is to detect if any intrusion activity exists in a packet. The detection engine employs Snort rules for this purpose. The rules are read into internal data structures or chains where they are matched against all packets. If a packet matches any rule, appropriate action is taken; otherwise the packet is dropped. Appropriate actions may be logging the packet or generating alerts.

The detection engine is the time-critical part of Snort. Depending upon how powerful your machine is and how many rules you have defined, it may take different amounts of time to respond to different packets. If traffic on your network is too high when Snort is working in NIDS mode, you may drop some packets and may not get a true real-time response. The load on the detection engine depends upon the following factors:

- Number of rules
- Power of the machine on which Snort is running
- Speed of internal bus used in the Snort machine
- Load on the network

When designing a Network Intrusion Detection System, you should keep all of these factors in mind.

Note that the detection system can dissect a packet and apply rules on different parts of the packet. These parts may be:

- The IP header of the packet.
- The Transport layer header. This header includes TCP, UDP or other transport layer headers. It may also work on the ICMP header.

- The application layer level header. Application layer headers include, but are not limited to, DNS header, FTP header, SNMP header, and SMTP header. You may have to use some indirect methods for application layer headers, like offset of data to be looked for.
- Packet payload. This means that you can create a rule that is used by the detection engine to find a string inside the data that is present inside the packet.

The detection engine works in different ways for different versions of Snort. In all 1.x versions of Snort, the detection engine stops further processing of a packet when a rule is matched. Depending upon the rule, the detection engine takes appropriate action by logging the packet or generating an alert. This means that if a packet matches criteria defined in multiple rules, only the first rule is applied to the packet without looking for other matches. This is fine except for one problem. A low priority rule generates a low priority alert, even if a high priority rule meriting a high priority alert is located later in the rule chain. This problem is rectified in Snort version 2 where all rules are matched against a packet before generating an alert. After matching all rules, the highest priority rule is selected to generate the alert.

The detection engine in Snort version 2.0 is completely rewritten so that it is a lot faster compared to detection in earlier versions of Snort. While Snort 2.0 is still not in release at the time of writing this book, earlier analysis shows that the new detection engine may be up to eighteen times faster.

1.3.4 Logging and Alerting System

Depending upon what the detection engine finds inside a packet, the packet may be used to log the activity or generate an alert. Logs are kept in simple text files, tcpdump-style files or some other form. All of the log files are stored under `/var/log/snort` folder by default. You can use `-l` command line options to modify the location of generating logs and alerts. Many command line options discussed in the next chapter can modify the type and detail of information that is logged by the logging and alerting system.

1.3.5 Output Modules

Output modules or plug-ins can do different operations depending on how you want to save output generated by the logging and alerting system of Snort. Basically these modules control the type of output generated by the logging and alerting system. Depending on the configuration, output modules can do things like the following:

- Simply logging to `/var/log/snort/alerts` file or some other file
- Sending SNMP traps
- Sending messages to syslog facility
- Logging to a database like MySQL or Oracle. You will learn more about using MySQL later in this book
- Generating eXtensible Markup Language (XML) output
- Modifying configuration on routers and firewalls.
- Sending Server Message Block (SMB) messages to Microsoft Windows-based machines

Other tools can also be used to send alerts in other formats such as e-mail messages or viewing alerts using a web interface. You will learn more about these in later chapters. Table 1-1 summarizes different components of an IDS.

Table 1-1 Components of an IDS

Name	Description
Packet Decoder	Prepares packets for processing.
Preprocessors or Input Plugins	Used to normalize protocol headers, detect anomalies, packet re-assembly and TCP stream re-assembly.
Detection Engine	Applies rules to packets.
Logging and Alerting System	Generates alert and log messages.
Output Modules	Process alerts and logs and generate final output.

1.4 Dealing with Switches

Depending upon the type of switches used, you can use Snort on a switch port. Some switches, like Cisco, allow you to replicate all ports traffic on one port where you can attach the Snort machine. These ports are usually referred to as spanning ports. The best place to install Snort is right behind the firewall or router so that all of the Internet traffic is visible to Snort before it enters any switch or hub. As an example, if you have a firewall with a T1 connection to the Internet and a switch is used on the inside, the typical connection scheme will be as shown in Figure 1-6.

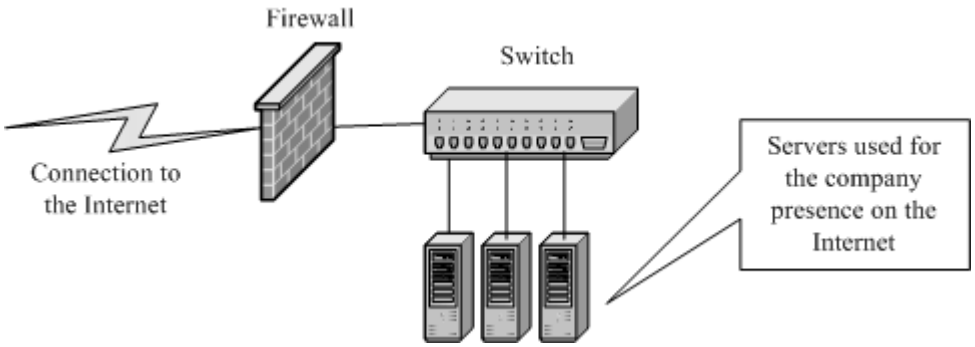

Figure 1-6 A typical connection scheme with one firewall and switched network.

If the switch you are using has a spanning port, you can connect the IDS machine to the spanning port as shown in Figure 1-7. All network traffic, including internal data flowing among company servers and the Internet data, will be visible to the IDS.

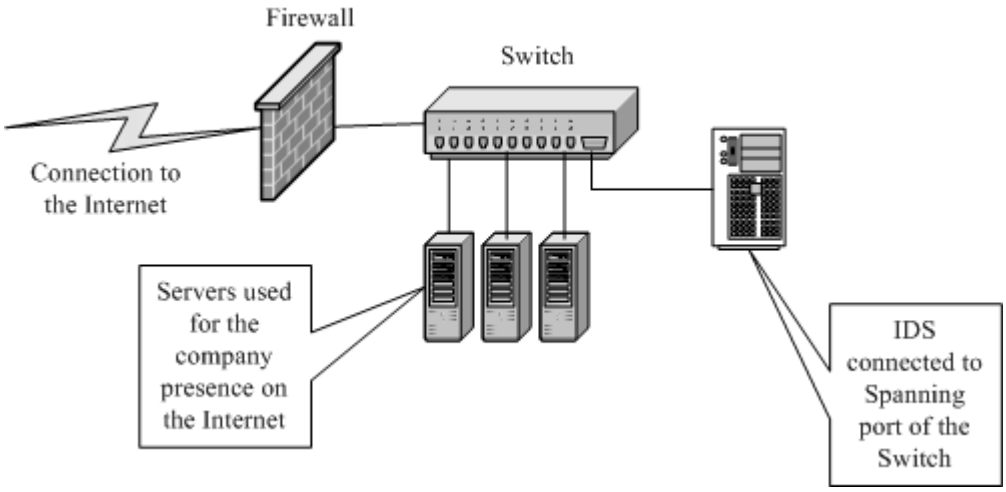

Figure 1-7 IDS connected a spanning port.

You can also connect the IDS to a small HUB or a Network TAP right behind the firewall, i.e., between firewall and the switch. In this case all incoming and outgoing traffic is visible to the IDS. The scheme is shown in Figure 1-8.

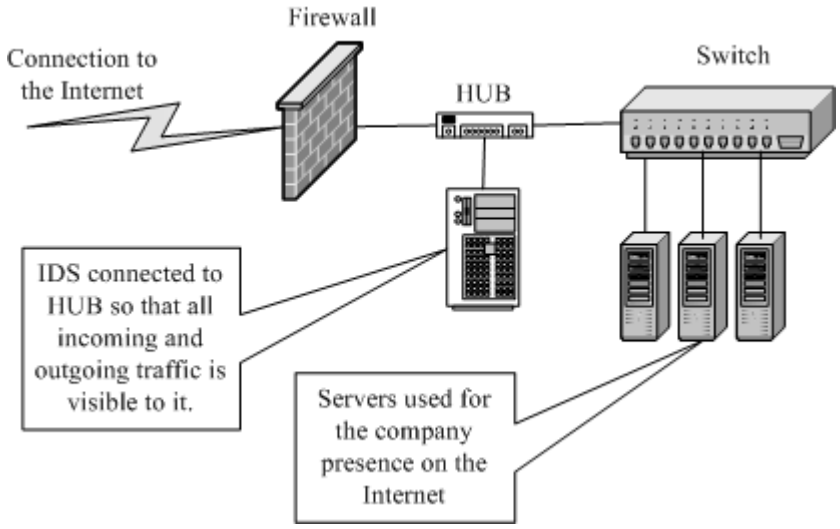

Figure 1-8 Connecting an IDS in a switched environment.

Note that when the IDS is connected as shown in Figure 1-8, data flowing among the company servers is not visible to the IDS. The IDS can see only that data which is coming from or going to the Internet. This is useful if you expect attacks from outside and the internal network is a trusted one.

1.5 TCP Stream Follow Up

A new preprocessor named Stream4 has been added to Snort. This preprocessor is capable of dealing with thousands of simultaneous streams and its configuration will be discussed in Chapter 4. It allows TCP stream reassembly and stateful inspection of TCP packets. This means that you can assemble packets in a particular TCP session to find anomalies and attacks that use multiple TCP packets. You can also look for packets coming to and/or originating from a particular server port.

1.6 Supported Platforms

Snort is supported on a number of hardware platforms and operating systems. Currently Snort is available for the following operating systems:

- Linux
- OpenBSD

- FreeBSD
- NetBSD
- Solaris (both Sparc and i386)
- HP-UX
- AIX
- IRIX
- MacOS
- Windows

For a current list of supported platforms, refer to the Snort home page at http://www.snort.org.

1.7 How to Protect IDS Itself

One major issue is how to protect the system on which your intrusion detection software is running. If security of the IDS is compromised, you may start getting false alarms or no alarms at all. The intruder may disable IDS before actually performing any attack. There are different ways to protect your system, starting from very general recommendations to some sophisticated methods. Some of these are mentioned below.

- The first thing that you can do is not to run any service on your IDS sensor itself. Network servers are the most common method of exploiting a system.
- New threats are discovered and patches are released by vendors. This is almost a continuous and non-stop process. The platform on which you are running IDS should be patched with the latest releases from your vendor. For example, if Snort is running on a Microsoft Windows machine, you should have all the latest security patches from Microsoft installed.
- Configure the IDS machine so that it does not respond to ping (ICMP Echo-type) packets.
- If you are running Snort on a Linux machine, use netfilter/iptable to block any unwanted data. Snort will still be able to see all of the data.
- You should use IDS only for the purpose of intrusion detection. It should not be used for other activities and user accounts should not be created except those that are absolutely necessary.

In addition to these common measures, Snort can be used in special cases as well. Following are two special techniques that can be used with Snort to protect it from being attacked.

1.7.1 Snort on Stealth Interface

You can run Snort on a stealth interface which only listens to the incoming traffic but does not send any data packets out. A special cable is used on the stealth interface. On the host where Snort is running, you have to short pins 1 and 2. Pins 3 and 6 are connected to same pins on the other side. Please see Snort FAQ at http://www.snort.org/docs/faq.html for more information on this arrangement.

1.7.2 Snort with no IP Address Interface

You can also use Snort on an interface where no IP address is assigned. For example, on a Linux machine, you can bring up interface `eth0` using command "`ifconfig eth0 up`" without assigning an actual IP address. The advantage is that when the Snort host doesn't have an IP address itself, nobody can access it. You can configure an IP address on eth1 that can be used to access the sensor itself. This is shown in Figure 1-9.

On Microsoft Windows systems, you can use an interface without binding TCP/IP to the interface, in which case no IP address will be assigned to the interface. Don't forget to disable other protocols and services on the interface as well. In some cases it has been noted that winpcap (library used on Microsoft Windows machines to capture packets) does not work well when no IP address is assigned on the interface. In such a case, you can use the following method.

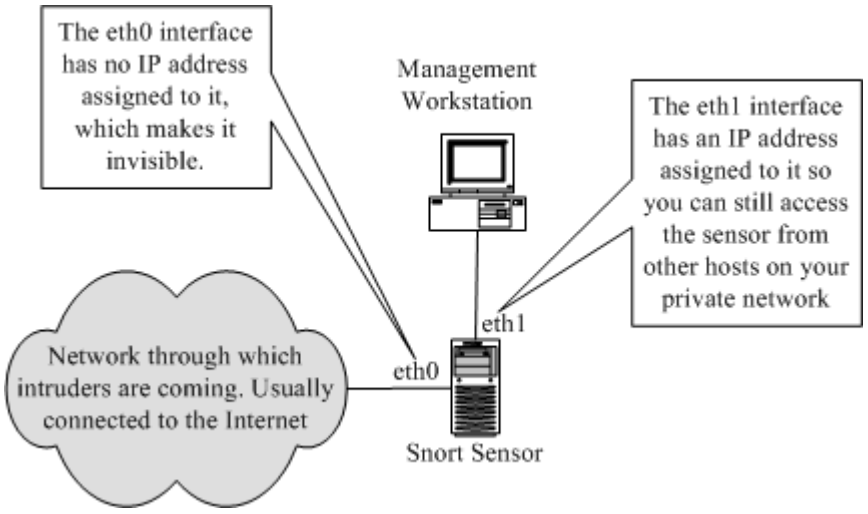

Figure 1-9 Snort sensor with two interfaces. One of these has no IP address assigned.

- Enable TCP/IP on the network interface that you want to use in the stealth mode. Disable everything other than TCP/IP.
- Enable DHCP client.
- Disable DHCP service.

This will cause no address to be assigned to the interface while the interface is still bound to TCP/IP networking.

1.8 References

1. Intrusion detection FAQ at http://www.sans.org/newlook/resources/IDFAQ/ID_FAQ.htm
2. Honey Pot Project at http://project.honeynet.org/
3. Snort FAQ at http://www.snort.org/docs/faq.html
4. Honeyd Honey Pot at http://www.citi.umich.edu/u/provos/honeyd/
5. Winpcap at http://winpcap.polito.it/
6. Cisco systems at http://www.cisco.com
7. Checkpoint web site at http://www.checkpoint.com
8. Netscreen at http://www.netscreen.com
9. Netfilter at http://www.netfilter.org
10. Snort at http://www.snort.org
11. The Nmap tool at http://www.nmap.org
12. Nessus at http://www.nessus.org
13. MySQL database at http://www.mysql.org
14. ACID at http://www.cert.org/kb/acid
15. Apache web server at http://www.apache.org