Avaya Solution & Interoperability Test Lab

Application Notes for Installing and Configuring Snort Inline for use with Avaya Voice over IP Telephony Services – Issue 1.0

Abstract

These Application Notes describe procedures that may be used to install and configure Snort Inline for use with Avaya Voice over IP (VoIP) telephony services. Both SIP and H.323 telephony services were tested for interoperability with Snort Inline. Snort is a popular, open source application that can passively monitor network traffic and act as an Intrusion Detection System (IDS) by generating alerts when threats are detected. When deployed in Inline mode, Snort will additionally act as an Intrusion Prevention System (IPS) by actively dropping the IP traffic associated with detected threats. Snort Inline was configured and installed on the Ubuntu Server distribution of Linux.

1. Introduction

These Application Notes describe procedures that may be used to install and configure Snort Inline on an Ubuntu Linux server for use with Avaya VoIP telephony services. In addition to documenting an example set of procedures, these Application Notes provide background information on Snort Inline and the use of Snort Inline. This will allow the procedures documented here to be more easily adapted to other environments (e.g. use of other Linux distributions).

From the www.snort.org website:

> "Snort is an open source network intrusion prevention and detection system utilizing a rule-driven language, which combines the benefits of signature, protocol and anomaly based inspection methods. With millions of downloads to date, Snort is the most widely deployed intrusion detection and prevention technology worldwide and has become the de facto standard for the industry."

Snort is used primarily to passively monitor network traffic and generate alerts when threats are detected. More recently, the Inline mode of deployment has become available and can be used to actively intercept and drop network traffic. The essence of Inline mode is that, a) Snort is configured and deployed on a server that forwards/routes network traffic as opposed to only sniffing network traffic and, b) Snort "alert" rules are changed into "drop" rules. "Drop" rules are specific to Snort Inline and will result in the dropping of packets that match the Snort rule.

Many Linux distributions include the iptables firewall application and Snort Inline interacts with iptables to receive and process network traffic. Appropriate iptables rules are used to direct network traffic to Snort Inline for inspection according to Snort rules. Given this interaction between Snort Inline and iptables, successful configuration of Snort Inline depends on successful configuration of iptables. Accordingly, these notes provide an example of an iptables ruleset that supports both integration with Snort Inline and interoperability with Avaya VoIP services.

The testing performed for these Application Notes covered both H.323 and SIP services as depicted in **Figure 1**.

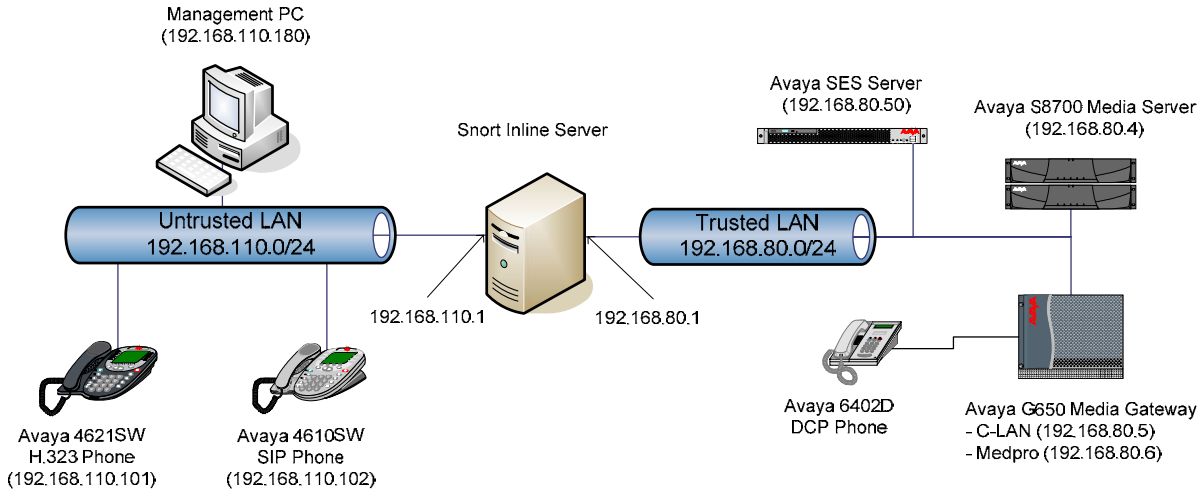

Figure 1 - Test Configuration

This configuration models an interior firewall deployment. Telephony endpoints and a management PC were located in the "untrusted" zone while Avaya Communication Manager and an Avaya SIP Enablement Services (SES) Server were located in the "trusted" zone. For this type of configuration, a primary objective is to allow only VoIP and management related traffic to flow from the untrusted zone to the trusted zone. This objective can be minimally met with the iptables firewall but this may leave systems vulnerable to higher level exploits that iptables cannot detect. For example, Avaya Communication Manager is managed with a web browser and the associated iptables rule permits HTTP traffic to Avaya Communication Manager. Unless additional measures are taken, this iptables rule will allow web server attacks to be directed at Avaya Communication Manager. The Snort Inline IDS/IPS can be used to provide better security by further inspecting the traffic that is permitted by this iptables rule and by blocking web server attacks found within this traffic.

In the configuration used for these Application Notes, the iptables application provided basic firewalling while Snort Inline provided higher level IDS/IPS functionality. An iptables ruleset was designed so that packets consistent with VoIP and management related traffic were passed to Snort Inline for processing according to Snort rules. Other traffic was blocked directly at the iptables level. This means that any traffic flowing between the untrusted and trusted zones must pass two levels of filtering, namely: the traffic must match iptables rules AND the traffic must pass Snort Inline processing.

The focus of these Application Notes is installation and configuration of the Snort Inline Server depicted in **Figure 1**. Procedures for installing and configuring Avaya VoIP services are not covered. The Ubuntu distribution of Linux was chosen for these Application Notes but Snort will run on a variety of Linux distributions.

It is assumed that:

- The reader has some level of familiarity with Linux and is comfortable with navigating a Linux shell.
- The reader has some level of familiarity with IP networking and the iptables firewall.
- The reader is comfortable with editing text files with a non-graphical editor such as "vi".

2. Equipment and Software Validated

The equipment and software/firmware used for these Application Notes is listed in **Table 1**.

Equipment	Software Tested
Avaya S8700 Media Server	R3.1 (R013x.01.0.628.5)
Avaya G650 with: TN2312BP IPSI TN799DP C-LAN TN2302AP IP Media Processor	 HW12 FW022 HW01 FW016 HW20 FW108
Avaya SES	R3.1 (SES-3.1.0.0-018.0)
Avaya 4621SW IP phone (H.323)	R2.2 (a20d01b2_2.bin)
Avaya 4610SW IP phone (SIP)	R2.2.2 (s10d01b2_2_2.bin)
Avaya 6402D DCP phone	NA
Ubuntu Linux	6.06 Server CD image: ubuntu-6.06.1-server-i386.iso
Snort	2.4.5
Snort Rules	VRT Certified Rules for Snort v2.4, registered user, current rules, downloaded on 9/1/2006, 14:16.
Oinkmaster	2.0

Table 1 - Equipment Tested

3. Configuration/Installation

3.1. Linux Installation

Since the installation of Linux is not central to these Application Notes, detailed steps for the installation of Ubuntu Server are not covered here. The installation steps beginning in **Section 3.2** assume that Ubuntu Server has already been successfully installed. These steps also assume that the Internet is accessible from the server so that Linux packages and source code bundles can be downloaded directly to the server.

Since Snort runs on a variety of Linux distributions, a distribution other than Ubuntu Server may be used. In this case, the steps presented in these Application Notes will need to be adapted to the particular Linux distribution that is used. The main prerequisite for running Snort Inline is that the server on which Linux is installed must have at least two network interfaces.

The use of Ubuntu Server will result in a fairly "bare bones" Linux installation (e.g. no graphical environment or graphical applications are installed). It is assumed that a simple text editor such as "vi" is used to edit text files when editing is required. Alternative text editors are available and may be used but installation and use of other editors is not covered here. The Linux shell commands used and the examples of file contents are presented in boxes with gray background such as the following example:

```
example command

example contents of file
```

Neither the Linux shell prompt nor responses from commands are depicted. It is assumed that commands are typed in then entered with the enter key.

The console interface of the PC was used exclusively for all installation and configuration tasks described in these Application Notes.

3.2. Installation of Prerequisite Linux Packages

This section describes the steps required to round out the base Ubuntu Server installation with packages that are prerequisites for items installed in subsequent sections. Although these steps are specific to Ubuntu Server Linux, information regarding prerequisite packages may be useful for installations on other Linux distributions.

Step 1 – Gain Access as Root User

All of the steps described in these Application Notes are performed as the root user so the first task is to supply a password for the root user. Login to a Linux shell using the user ID and password that were created during the installation of Linux. Then create a password for the root user with the following command:

```
sudo passwd root
```

Enter the password for the root user (twice) when prompted. Then logout with the **`exit`** command and log back in to a Linux shell as the root user with the root password just created.

Step 2 – Enable Internet Package Archives

In later steps, the system will be updated and several packages will be retrieved from the Internet. To support this, the `sources.list` configuration file must be edited to enable Internet based package archives and to eliminate the CD-ROM as a source of packages. Edit the `sources.list` configuration file with the following command:

```
vi /etc/apt/sources.list
```

In the `sources.list` file, insert a "#' character to comment out the line starting with "deb cdrom". Also, delete the "#' character to uncomment at least one of the lines which specifies an Internet archive. The resulting file will be as follows (the edited lines are highlighted in bold):

```
# deb cdrom:[Ubuntu-Server 6.06 _Dapper Drake_ - Release i386 (20060531)]/
dapper main restricted

deb http://us.archive.ubuntu.com/ubuntu/ dapper main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ dapper main restricted
```

Step 3 – Update the System

The **apt-get** program is the main tool used within Ubuntu Linux for package management (i.e. installing and removing software). Use the following two commands to update the list of available packages and then to upgrade installed packages to the latest available versions:

```
apt-get update
apt-get upgrade
```

Type "Y" and hit enter in response to the "Do you want to continue" question presented after running **apt-get upgrade.** The system will download and install upgrades as required.

Step 4 – Install Development Packages

Because several source code distributions will be compiled in later steps, suitable development tools and packages must be installed. Use the following command, entered as one line, to retrieve and install the required packages:

```
apt-get install autoconf automake1.9 autotools-dev binutils bison cpp cpp-4.0
flex gcc gcc-4.0 libc6-dev linux-kernel-headers m4 make perl perl-modules
```

The system will retrieve and install all 16 packages.

Step 5 – Install iptables and Perl Related Packages

To interface with iptables, Snort Inline requires a complete iptables installation including the `iptables-dev` package. Additional Perl libraries and packages are required by the Oinkmaster tool discussed in **Section 3.4**. Use the following command, entered as one line, to retrieve and install the required packages:

```
apt-get install iptables iptables-dev libpcre3 libpcre3-dev libpcrecpp0
libpcap0.8 libpcap0.8-dev
```

The system will retrieve and install 5 of the 7 packages. The `iptables and libpcap0.8 packages` will have already been installed and will therefore not be retrieved. These two packages are identified here, however, in case the information is useful when other Linux distributions are used.

Step 6 – Configure HTTP Proxy (Optional)

The **wget** command will be used to fetch source code bundles in later steps. If an HTTP proxy server is used, it will need to be identified in a **.wgetrc** file for use by **wget**. If so, use a command similar to following with appropriate substitutions for the proxy server IP address and port. This step should be omitted if an HTTP proxy server is not required.

```
echo "http_proxy=192.168.15.12:8000" > /root/.wgetrc
```

3.3. Installation of Snort Inline

Step 1 – Install Libnet

Libnet is a networking library required by Snort Inline. The version of Snort Inline used for these Application Notes requires a 1.0.x version of Libnet and the 1.0.2a version of Libnet was chosen and used here. Since this older version of Libnet is now obsolete, it is no longer available as a binary package via **apt-get** package management. Hence, the source code must be obtained and compiled locally. Use the following commands to retrieve, compile, and install the source code for the 1.0.2a version of Libnet:

```
cd /usr/src
wget http://www.packetfactory.net/libnet/dist/deprecated/libnet-1.0.2a.tar.gz
tar xzvf libnet-1.0.2a.tar.gz
cd Libnet-1.0.2a
./configure
make
make install
```

Step 2 – Install Snort Inline

Two different distributions of Snort are currently available. A distribution specifically geared towards inline mode is available at http://snort-inline.sourceforge.net/ and is referred to as snort_inline. At one point, this was the only distribution of Snort that supported inline mode. As of Snort 2.3.0 RC1, however, the inline mode was incorporated into the official Snort project. Hence, the snort.org distribution is also now capable of supporting inline mode and this distribution was used for the testing here. The snort_inline distribution at http://snort-inline.sourceforge.net continues to be available as the developers there continue to provide enhancements for inline mode. As of this writing, the main features supported by the distribution at http://snort-inline.sourceforge.net but not supported by the distribution at snort.org are:

- Support for the new NFQUEUE target allowing multiple QUEUE targets and therefore multiple instances of snort_inline.
- Support for stream reassembly in inline mode which affords protection against session splicing attacks.
- Support for sticky-drop i.e. dropping all traffic from an attacker for an extended period of time.
- Support for scanning of traffic for viruses with clamav (clamav is claimed to be a resource hog).

The source code for Snort must be obtained and compiled locally. This is because the version of Snort retrieved using **`apt-get`** package management is a "vanilla" version that does not support inline mode. Use the following commands to retrieve, configure, compile, and install the Snort source code. The **`--enable-inline`** configuration script option is required to enable inline mode.

```
cd /usr/src
wget http://www.snort.org/dl/current/snort-2.4.5.tar.gz
tar xzvf snort-2.4.5.tar.gz
cd snort-2.4.5
./configure --enable-inline --with-libipq-includes=/usr/include/libipq
make
make install
```

Step 3 – Configure the Snort Inline Installation

Use the following commands to create directories for the Snort configuration files, rules files, and log files and to copy the Snort configuration files to the desired directory:

```
mkdir /var/log/snort
mkdir /etc/snort
mkdir /etc/snort/rules
cp /usr/src/snort-2.4.5/etc/* /etc/snort
```

Edit the `snort.conf` file with the following command:

```
vi /etc/snort/snort.conf
```

Modify the `snort.conf` file so that the RULE_PATH variable references the new Snort rules directory. The resulting file will be as follows (the edited line is highlighted in bold):

```
# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
```

The installation of Snort Inline itself is now complete but Snort has no rules to work with. The downloading of rules is a distinct process and is handled by the Oinkmaster tool which is covered in the next section.

3.4. Installation of Oinkmaster

Oinkmaster is a Perl script used to update and manage Snort rules. The snort.org site describes the various Snort rulesets that are available for download. The Sourcefire VRT Certified Rules available to a registered snort.org user are used here.

Step 1 – Generate an Oink Code

The retrieval of rules via the Oinkmaster tool requires the use of an "Oink Code". An Oink Code is available by registering and creating an account at snort.org. After creating and logging into an account at snort.org, generate a Snort Oink Code by clicking the "Get Code" button at the bottom of the "Account Settings" page. The Oink Code generated will be used during the editing of the `oinkmaster.conf` file in **Step 3** in this section.

Step 2 – Install Oinkmaster

Retrieve and set up the Oinkmaster tool using the commands below. These commands will:

- Retrieve and unpack the Oinkmaster 2.0 distribution which is the latest version of Oinkmaster as of this writing. The URL used assumes that the Oinkmaster source package is available at the "easynews" download mirror.
- Copy the `oinkmaster.pl` script file to the `/usr/local/bin` directory so that it will be found in the executable path.
- Copy the `oinkmaster.conf` configuration file to `/etc` which is where the `oinkmaster.pl` script will look for it by default.

```
cd /usr/src
wget http://easynews.dl.sourceforge.net/sourceforge/oinkmaster/oinkmaster-
2.0.tar.gz
tar xzvf oinkmaster-2.0.tar.gz
cd oinkmaster-2.0
cp oinkmaster.pl /usr/local/bin
cp oinkmaster.conf /etc
```

Step 3 – Edit the Oinkmaster Configuration File

Edit the oinkmaster.conf file with the following command:

```
vi /etc/oinkmaster.conf
```

Modify the `oinkmaster.conf` file to change the line which controls retrieval of a rules snapshot for the Snort 2.4 ruleset. Insert the Oink Code generated in **Step 1** in this section and uncomment the line by deleting the leading "#" character. The resulting file will appear as below (the edited line is highlighted in bold). The Oink Code used in the example below is NOT an actual Oink Code. An actual Oink Code, generated at the snort.org site, should be used instead.

```
# URL examples follows. Replace <oinkcode> with the code you get on the
# Snort site in your registered user profile.

# Example for Snort 2.4
url = http://www.snort.org/pub-
bin/oinkmaster.cgi/234ef981dab34964db36f1f209fed1f5b7891a3/snortrules-
snapshot-2.4.tar.gz

# Example for Snort-current ("current" means cvs snapshots).
# url = http://www.snort.org/pub-bin/oinkmaster.cgi/<oinkcode>/snortrules-
snapshot-CURRENT.tar.gz
```

One additional change to the `oinkmaster.conf` file is required. This change will instruct the Oinkmaster script to convert Snort "alert" rules to "drop" rules. Each Snort rule has an associated Snort ID (sometimes referred to as a Signature ID) or SID. Using the "*" character as a wildcard to match on all SID's, the Oinkmaster script will be instructed to convert all "alert" rules to "drop" rules. Modify the `oinkmaster.conf` file to insert this instruction. The resulting file will appear as follows (the inserted lines are highlighted in bold):

```
# Example to add "tag" stuff to SID 1325.
# modifysid 1325 "sid:1325;" | "sid:1325; tag: host, src, 300, seconds;"

# Example to make SID 1378 a 'drop' rule (valid if you're running
# Snort_inline).
# modifysid 1378 "^alert" | "drop"
#
# Use example above but change ALL alert rules to drop rules
modifysid * "^alert" | "drop"
#
# Example to replace first occurrence of $EXTERNAL_NET with $HOME_NET
# in SID 302.
# modifysid 302 "\$EXTERNAL_NET" | "\$HOME_NET"
```

Step 4 – Run the Oinkmaster Script

The Oinkmaster script will download and unpack the official Snort ruleset and process the rules. Run the Oinkmaster script with the command below. The `-o` option is used to identify the Snort rules directory. Before running the script, examine the `/etc/snort/rules` directory to observe that there are no rules files. After running the script, examine the `/etc/snort/rules` directory to observe that rules files have been created. Also, view one or more of the rules files (e.g. `mysql.rules`) to observe that "drop" rules have been created.

```
oinkmaster.pl -o /etc/snort/rules
```

Step 5 – Create local.rules File

The `snort.conf` file instructs Snort as to what rules files are to be expected. By default, the `snort.conf` file includes a reference to the file: `local.rules`. This file is intended to contain rules created locally by end users. Hence, by default, Snort will look for and assume that there is a `local.rules` file in the rules directory. In order to start Snort successfully, the `local.rules` file must exist even if empty. Use the **`touch`** command as follows to create an empty `local.rules` file:

```
touch /etc/snort/rules/local.rules
```

A potential alternative to the above is to modify the `snort.conf` file so that a `local.rules` file is not referenced (e.g. delete or comment out the line in the `snort.conf` file that refers to the `local.rules` file). However, care must be taken with this alternative approach so as to avoid inadvertently missing local rules if they are created later and placed in the `local.rules` file.

3.5. Networking Configuration

Two main tasks are required in order to finalize the configuration of networking related information:

1. Ensure that both network interfaces are configured appropriately.
2. Enable packet forwarding/routing.

Step 1 – Configure Network Interfaces

Edit the network interface properties file with the following command:

```
vi /etc/network/interfaces
```

Modify the network interface properties file to reflect the networking environment where the server is installed. The resulting file should be similar to the example file shown below. The example file shown below reflects the following:

- The lines for the second network interface (eth1) were added to the existing file. The Ubuntu Server Linux installation program configured only the loopback and primary network interface (eth0).
- The subnets depicted in **Figure 1** were used and the parameters in the example file are consistent with those.
- The default gateway and DNS parameters for the primary network interface were commented out with a "#" character. These parameters were configured by the Ubuntu Server Linux installation program but were not needed for the final networking configuration.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
      address 192.168.110.1
      netmask 255.255.255.0
      network 192.168.110.0
      broadcast 192.168.110.255
      # gateway 192.168.110.4
      # dns-* options are implemented by the resolvconf package, if installed
      # dns-nameservers 192.168.6.11 192.168.7.12

# The secondary network interface
auto eth1
iface eth1 inet static
      address 192.168.80.1
      netmask 255.255.255.0
      network 192.168.80.0
      broadcast 192.168.80.255
```

Step 2 – Enable Packet Forwarding

In order to enable packet forwarding from one network interface to another, edit the `sysctl.conf` file with the following command:

```
vi /etc/sysctl.conf
```

Modify the `sysctl.conf` file to uncomment the line related to ipv4 packet forwarding by removing the leading "#" character. The resulting file will appear as follows (the edited line is highlighted in bold):

```
# Uncomment the next line to enable TCP/IP SYN cookies
#net/ipv4/tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net/ipv4/ip_forward=1
```

Step 3 – Reboot the System

In order to activate the networking changes performed is **Step 1** and **Step 2** in this section, reboot the system using the **`reboot`** command then log back in as the root user after the system has rebooted.

The server will now be able to forward IP packets from one network interface to the other thus behaving as a router. No static routes have been added nor have any routing protocols been installed or activated. Hence, given the simple configuration in **Figure 1**, the server will only be able to act as a router for endpoints residing in the two subnets that the server is directly connected to.

Although the details for VoIP telephony configuration are not covered in these Application Notes, it should be noted that the Ubuntu server network interface IP addresses were used as follows (refer to **Figure 1**):

- The default gateway for the telephone endpoints and management PC was the IP address of the primary network interface on the Ubuntu server (i.e. 192.168.110.1).
- The default gateway for Avaya Communication Manger, Avaya SES server, C-LAN board, and the Medpro board was the IP address of the of the secondary network interface Ubuntu server (i.e. 192.168.80.1).

3.6. Running Snort Inline

Three final steps are required to get Snort Inline up and running:

1. Load the `ip_queue` module. This is the module used to allow Snort to intercept and process IP packets. The `ip_queue` module works in conjunction with the iptables QUEUE target. Packets that are directed to the iptables QUEUE target are then intercepted by Snort Inline via the `ip_queue` module.
2. Configure an iptables ruleset with two main objectives:
    a. Use iptables ACCEPT and DROP rules to create a base packet filtering firewall identifying packets that are to be accepted or dropped at the iptables level.
    b. Use the iptables QUEUE target to specify traffic that is to be passed to Snort for further processing.
3. Start the Snort process.

Note that these three final steps will need to be repeated after any system reboot. All other prior configuration steps will survive a reboot but these final three steps will not. In order to make it easier to bring the Snort IDS/IPS up after a reboot, it may be desirable to use a script. A simple example shell script is provided in **Section 3.7**.

Since the objective is to apply Snort processing to IP packets that are ***forwarded*** from one network interface to another, the iptables FORWARD chain will be used for iptables rules. Note that use of other iptables chains is also possible but is not covered here.

In the example iptables rules provided here, the rules only identify traffic related to H.323, SIP, web browser management, and SSH management. All other types of traffic are therefore dropped at the iptables level. Furthermore, all traffic that matches an iptables rule is passed on to Snort Inline for further IDS/IPS filtering. In other words, there are no iptables ACCEPT rules. Rather, all iptables rules that identify a traffic type use the QUEUE target. This means that all packets matching an iptables rule will be passed to Snort Inline.

Step 1 – Load the ip_queue Module

Load the ip_queue module with the following command:

```
modprobe ip_queue
```

If desired, the following command may be used to check that the ip_queue module has been loaded properly. The output of this command will contain a line listing the ip_queue module if the ip_queue module is running.

```
lsmod | grep ip_queue
```

Step 2 – Add iptables Rules

Prior to adding new rules to the iptables FORWARD chain, flush any existing rules and set the default policy with the commands shown below. The first of these commands flushes any rules that might already exist in the iptables FORWARD chain. The second of these commands sets the default policy of the FORWARD chain to DROP. This means that any packet type not matching a rule will be dropped at the iptables level.

```
iptables -F FORWARD
iptables -P FORWARD DROP
```

Add desired rules to the FORWARD chain using the QUEUE target with the commands shown below. Specific reasons for the rules depicted below can be found in the comments in the shell script file presented in **Section 3.7**. The iptables command line options used below are as follows:

- `-A`: add rule
- `-s`: source address
- `-d`: destination address
- `-p`: protocol
- `--dport`: destination port
- `--sport`: source port

```
iptables -A FORWARD -s 192.168.110.0/24 -d 192.168.80.5 -p udp --dport 1719 -j QUEUE
iptables -A FORWARD -s 192.168.80.5 -d 192.168.110.0/24 -p udp --sport 1719 -j QUEUE
iptables -A FORWARD -s 192.168.110.0/24 -d 192.168.80.5 -p tcp --dport 1720 -j QUEUE
iptables -A FORWARD -s 192.168.80.5 -d 192.168.110.0/24 -p tcp --sport 1720 -j QUEUE
iptables -A FORWARD -s 192.168.110.0/24 -d 192.168.80.6 -p udp --dport 2048:4327 -j
QUEUE
iptables -A FORWARD -s 192.168.80.6 -d 192.168.110.0/24 -p udp --sport 2048:4327 -j
QUEUE
iptables -A FORWARD -s 192.168.110.0/24 -d 192.168.80.50 -p udp --dport 5060 -j QUEUE
iptables -A FORWARD -s 192.168.80.50 -d 192.168.110.0/24 -p udp --dport 5060 -j QUEUE
iptables -A FORWARD -s 192.168.110.180 -d 192.168.80.4 -p tcp --dport 22 -j QUEUE
iptables -A FORWARD -s 192.168.80.4 -d 192.168.110.180 -p tcp --sport 22 -j QUEUE
iptables -A FORWARD -s 192.168.110.180 -d 192.168.80.4 -p tcp --dport 80 -j QUEUE
iptables -A FORWARD -s 192.168.80.4 -d 192.168.110.180 -p tcp --sport 80 -j QUEUE
iptables -A FORWARD -s 192.168.110.180 -d 192.168.80.4 -p tcp --dport 443 -j QUEUE
iptables -A FORWARD -s 192.168.80.4 -d 192.168.110.180 -p tcp --sport 443 -j QUEUE
iptables -A FORWARD -s 192.168.110.180 -d 192.168.80.50 -p tcp --dport 80 -j QUEUE
iptables -A FORWARD -s 192.168.80.50 -d 192.168.110.180 -p tcp --sport 80 -j QUEUE
iptables -A FORWARD -s 192.168.110.180 -d 192.168.80.50 -p tcp --dport 443 -j QUEUE
iptables -A FORWARD -s 192.168.80.50 -d 192.168.110.180 -p tcp --sport 443 -j QUEUE
```

Use of the QUEUE target in the above rules is the means by which iptables is instructed to pass packets matching the rule to the ip_queue module. From the ip_queue module, these packets will then be intercepted by Snort Inline. If a packet is dropped within Snort Inline due to a Snort drop rule, then the packet will not be forwarded.

Step 3 – Start Snort Inline

Start the Snort process using the command shown below. The command is entered all as one line and the command line options are as follows:

- `-c`: identify the snort.conf file
- `-Q`: get packets from iptables (i.e. run Snort in inline mode)
- `-N`: disable packet logging (alerts for dropped packets are still logged, but contents of all processed packets are not logged)
- `-l`: identify the log file directory
- `-t`: chroot to the identified directory
- `-D`: run in daemon mode (i.e. run in background and free up terminal session)

```
/usr/local/bin/snort -c /etc/snort/snort.conf -Q -N -l /var/log/snort -t
/var/log/snort -D
```

Several startup messages will be reported. The following text is displayed when Snort has started successfully:

```
--== Initialisation Complete ==--
```

Snort Inline is now running and able to process packets.

If desired, the procedures listed below can be used to "undo" the steps presented in this section. Note that these procedures are optional and will not normally be needed.

1. Find the process ID for Snort via the following: **ps aux | grep snort**
2. Kill the process ID for Snort found in 1 above (e.g. kill the snort process of 1234): **kill 1234**
3. Flush the iptables FORWARD chain: **iptables -F FORWARD**
4. Return the default policy for the iptables FORWARD chain to ACCEPT: **iptables -P FORWARD ACCEPT**
5. Unload the ip_queue module: **rmmod ip_queue**

3.7. Example Shell Script

The example shell script shown below performs the equivalent of the configuration steps covered in **Section 3.6**. These are the steps that will not survive a reboot of the IDS/IPS server and may therefore be more conveniently handled via such a script.

```
#!/bin/bash
#
# Simple script for setting up iptables and starting snort inline
#
# Configuration assumptions:
# - WARNING: This script flushes all rules from the iptables
#   FORWARD chain.  It is assumed that the iptables FORWARD chain
#   is used exclusively for passing traffic to Snort Inline.
# - snort executable: /usr/local/bin/snort
# - snort configuration file: /etc/snort/snort.conf
# - snort logging directory: /var/log/snort
# - 192.168.110.0/24 is subnet for H.323 and SIP phones and a management PC
# - 192.168.80.0/24 is subnet for common telephony equipment.
# - The Linux server routes between the above two subnets and
#   acts as an iptables firewall and Snort based IDS/IPS.
# - Management PC at 192.168.110.180
# - Avaya Communication Manager management interface at 192.168.80.4
# - CLAN at 192.168.80.5
# - Medpro at 192.168.80.6
# - Avaya SES at 192.168.80.50

# Load ip_queue module
/sbin/modprobe ip_queue


# Flush rules from FORWARD chain
iptables -F FORWARD

# Set default policy for FORWARD chain as DROP
iptables -P FORWARD DROP

# Add rules to FORWARD chain.  The QUEUE target is used to
# pass allowed traffic to the ip_queue module for subsequent
# processing by snort.

# Allow UDP port 1719 to and from Avaya CLAN board for H.323 phone signaling
iptables -A FORWARD -s 192.168.110.0/24 -d 192.168.80.5 -p udp --dport 1719 -j QUEUE
iptables -A FORWARD -s 192.168.80.5 -d 192.168.110.0/24 -p udp --sport 1719 -j QUEUE

# Allow TCP port 1720 to and from Avaya CLAN board for H.323 phone signaling
iptables -A FORWARD -s 192.168.110.0/24 -d 192.168.80.5 -p tcp --dport 1720 -j QUEUE
iptables -A FORWARD -s 192.168.80.5 -d 192.168.110.0/24 -p tcp --sport 1720 -j QUEUE

# Allow UDP ports to and from Avaya Medpro board for RTP/RTCP traffic.
# The specific UDP ports allowed should match the UDP port settings on the
# Avaya Communication Manager network region form.
iptables -A FORWARD -s 192.168.110.0/24 -d 192.168.80.6 -p udp --dport 2048:4327 -j
QUEUE
iptables -A FORWARD -s 192.168.80.6 -d 192.168.110.0/24 -p udp --sport 2048:4327 -j
QUEUE
```

```
# Allow UDP port 5060 to and from Avaya SES for SIP phone signaling
iptables -A FORWARD -s 192.168.110.0/24 -d 192.168.80.50 -p udp --dport 5060 -j QUEUE
iptables -A FORWARD -s 192.168.80.50 -d 192.168.110.0/24 -p udp --dport 5060 -j QUEUE

# Allow SSH, HTTP, and HTTPS connections to Avaya Communication Manager from
management PC.
iptables -A FORWARD -s 192.168.110.180 -d 192.168.80.4 -p tcp --dport 22 -j QUEUE
iptables -A FORWARD -s 192.168.80.4 -d 192.168.110.180 -p tcp --sport 22 -j QUEUE
iptables -A FORWARD -s 192.168.110.180 -d 192.168.80.4 -p tcp --dport 80 -j QUEUE
iptables -A FORWARD -s 192.168.80.4 -d 192.168.110.180 -p tcp --sport 80 -j QUEUE
iptables -A FORWARD -s 192.168.110.180 -d 192.168.80.4 -p tcp --dport 443 -j QUEUE
iptables -A FORWARD -s 192.168.80.4 -d 192.168.110.180 -p tcp --sport 443 -j QUEUE

# Allow HTTP and HTTPS connections to Avaya SIP server from management PC.
iptables -A FORWARD -s 192.168.110.180 -d 192.168.80.50 -p tcp --dport 80 -j QUEUE
iptables -A FORWARD -s 192.168.80.50 -d 192.168.110.180 -p tcp --sport 80 -j QUEUE
iptables -A FORWARD -s 192.168.110.180 -d 192.168.80.50 -p tcp --dport 443 -j QUEUE
iptables -A FORWARD -s 192.168.80.50 -d 192.168.110.180 -p tcp --sport 443 -j QUEUE

# Start snort.
/usr/local/bin/snort -c /etc/snort/snort.conf -Q -N -l /var/log/snort -t
/var/log/snort -D
```

4. Verification Steps

This section contains tests that may be performed to ensure that the Snort Inline server has been installed and configured properly. The objectives of these tests are to:

- Verify that the base set of iptables rules permit VoIP and management related traffic but do not permit other types of traffic (Steps 1 to 14 below).
- Verify that Snort Inline does not interfere with VoIP and management related traffic (Steps 1 to 14 below).
- Verify that Snort Inline does not allow known threats to be forwarded from the untrusted zone to the trusted zone (Steps 15 and 16 below).

All of the verification steps presented here were successfully executed during the creation of these Application Notes.

Verification Steps

1. Verify that a SIP phone can register to the SES server.
2. Verify that an H.323 phone can register to the C-LAN.
3. Place a test call between a SIP phone and a wired DCP phone: Verify that audio paths are established in both directions.
4. Place a test call between an H.323 phone and a wired DCP phone: Verify that audio paths are established in both directions.
5. Place a test call between an H.323 phone and a SIP phone: Verify that audio paths are established in both directions.
6. Verify that an SSH session can be established from the management PC to Avaya Communication Manger.
7. Verify that an HTTP/HTTPS browser session can be established from the management PC to Avaya Communication Manager.
8. Verify that an HTTP/HTTPS browser session can be established from the management PC to the Avaya SES Server.
9. Verify that a telnet session from the management PC to Avaya Communication Manager cannot be established. Note that the iptables rules only permitted SSH connectivity to Avaya Communication Manager.
10. Verify that an SSH session from the management PC to the Avaya SES Server cannot be established.
11. Verify that a Telnet session from the management PC to the Avaya SES server cannot be established.
12. Verify that an HTTP/HTTPS session from the management PC to an additional web server in the trusted zone cannot be established.
13. Verify that an ICMP echo request (ping) initiated from the management PC is not forwarded to Avaya Communication Manager.
14. Verify that an ICMP echo request (ping) initiated from the management PC is not forwarded to the Avaya SES server.

15. Verify that an attempt to request a URL of the form "http://192.168.80.4/php.cgi" is blocked. Enter a URL of this form in the web browser at the management PC using the IP address of Avaya Communication Manager. Verify that the attempt to connect to this URL is blocked and verify that the attempt is logged in the Snort log file (`/var/log/snort/alert`). Note: the "php.cgi" string is identified as a threat in Snort SID 824 in the `web-cgi.rules` file.
16. Verify that an attempt to request a URL of the form "http://192.168.80.4/php.exe" is blocked. Enter a URL of this form in the web browser at the management PC using the IP address of Avaya Communication Manager. Verify that the attempt to connect to this URL is blocked and verify that the attempt is logged in the Snort log file (`/var/log/snort/alert`). Note: the "php.exe" string is identified as a threat in Snort SID 1773 in the `web-php.rules` file.

5. Conclusion

These Application Notes supply the steps needed to install and configure the Snort Inline IDS/IPS application for use with Avaya VoIP services. The verification steps, although not exhaustive, demonstrated that Snort Inline and the iptables firewall were correctly deployed in the test environment and were receiving and processing traffic as expected. To the extent that VoIP and management related connections were verified, Snort Inline did not interfere with the associated traffic.

The following areas may be of further interest as potential enhancements to the configuration and verification steps presented in these Application Notes:

- Snort rules are updated over time as new threats are identified. It may be desirable to run the Oinkmaster tool as a cron job so that rules are updated periodically and automatically.
- It may be preferable to start Snort from an init.d script so that it is automatically launched after a system restart. The commands covered in **Section 3.6** would be the ones to consider for coverage in such a script. Robust designs for these types of scripts tend to be dependent on the Linux distribution used. An example for Redhat/Fedora is given in: http://linuxgazette.net/118/savage.html
- The verification steps did not include any load or performance testing. It is not known how well Snort Inline would perform as the number of simultaneous VoIP sessions is increased.

6. Additional References

1. The installation of Snort presented here would be considered a bare minimum installation. Various other associated applications may be installed (e.g. databases and web interfaces for access to Snort reporting). Suggestions can be found at http://www.snort.org. This site provides documentation, discussion groups, and various installation "cookbooks" for different environments.
2. The alternative snort_inline distribution can be found at: http://snort-inline.sourceforge.net. William Metcalf is associated with this project and provided the information regarding snort_inline described in **Section 3.3.**
3. The following article (two parts) was quite helpful in creating these applications notes:
   http://linuxgazette.net/117/savage.html
   http://linuxgazette.net/118/savage.html
   Note that the above article was written before the inline mode of Snort was incorporated into the official project, does not cover use of the iptables FORWARD chain, and made use of a Red Hat/Fedora distribution of Linux.
4. Further general information on Oinkmaster can be found at http://oinkmaster.sourceforge.net. An alternative set of installation instructions for Oinkmaster can be found at: http://snort.org/docs/setup_guides/Installing_and_configuring_OinkMaster.pdf.

Please e-mail any questions or comments pertaining to these Application Notes along with the full title name and filename, located in the lower right corner, directly to the Avaya Solution & Interoperability Test Lab at interoplabnotes@list.avaya.com.