P-Series Installation and Operation Guide

Version 1.4.3 July 2006 PN: 100-00044-00

FORCE10™

Copyright 2006 Force10 Networks

Trademarks

Statement of Conditions

In the interest of improving internal design, operational function, and/or reliability, Force10 Networks reserves the right to make changes to products described in this document without notice. Force10 Networks does not assume any liability that may occur due to the use or application of the product(s) described herein.

USA Federal Communications Commission (FCC) Statement

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC rules. These limits are designated to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy. If it is not installed and used in accordance to the instructions, it may cause harmful interference to radio communications.

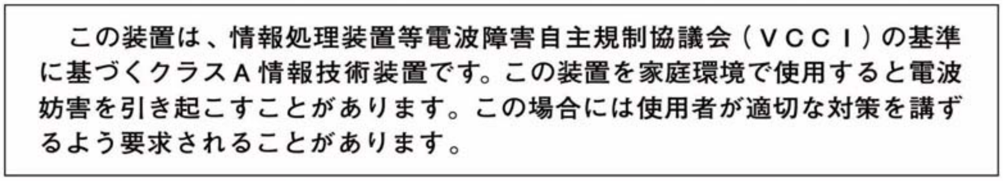
Operation of this equipment in a residential area is likely to cause harmful interference, in which case users will be required to take whatever measures necessary to correct the interference at their own expense.

Canadian Department of Communication Statement

The digital apparatus does not exceed the Class A limits for radio noise emissions from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications.

Attention: Le present appareil numerique n' emet pas de perturbations radioelectriques depassant les normes applicables aux appareils numeriques de la Class A prescrites dans le Reglement sur les interferences radioelectriques etabli par le ministere des Communications du Canada.

VCCI Compliance for Class A Equipment (Japan)

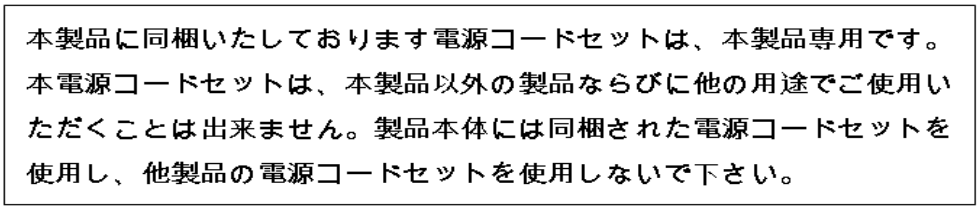


この装置は、情報処理装置等電波障害自主規制協議会（ＶＣＣＩ）の基準に基づくクラスＡ情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。

This is Class A product based on the standard of the Voluntary Control Council For Interference by Information Technology Equipment (VCCI). If this equipment is used in a domestic environment, radio disturbance may arise. When such trouble occurs, the user may be required to take corrective actions.\

Caution: This device is a Class A product. In a domestic environment, this device can cause radio interference, in which case, the user may be required to take appropriate measures.

Danger: AC Power cords are for use with Force10 Networks equipment only, do not use Force10 Networks AC Power cords with any unauthorized hardware.



本製品に同梱いたしております電源コードセットは、本製品専用です。本電源コードセットは、本製品以外の製品ならびに他の用途でご使用いただくことは出来ません。製品本体には同梱された電源コードセットを使用し、他製品の電源コードセットを使用しないで下さい。

Contents

Preface

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Appendix A

Appendix B

List of Figures

List of Tables

Preface About this Guide

Objectives

This document provides installation and operation instructions for the P-Series appliances.

Audience

This guide is intended to be used by network engineers. The P-Series appliances are Unix-based products which run rule management software based on Fedora Core 3 and FreeBSD 6.0. As such, understanding how to operate the appliances requires a basic knowledge of Unix, including the *vi* editor.

Conventions

This document uses the following conventions to describe command syntax:

Convention	Description
keyword	Keywords are in bold and should be entered at the command prompt as listed.
parameter	Parameters are in italics and require a number or word to be entered at the command prompt.
{X}	Keywords and parameters within braces must be entered at the command prompt.
[X]	Keywords and parameters within brackets are optional.
x|y	Keywords and parameters separated by a bar require you to choose one.

Technical Support

Information on operating the appliance can be accessed through manual pages (man pages) with the command **man** *command*. The command **man meta** displays the man pages on the command line interface; and **man mtp** displays them on the *Ncurses* interface. Man pages for the compiler can be accessed with **man mtp-compiler**.

- For information on Snort or creating Snort rules, visit www.snort.org.
- For information on Unix commands and the *vi* editor, see Appendix A, on page 59.

Related Documents

- *P-Series Release Notes*

Chapter 1 Installation

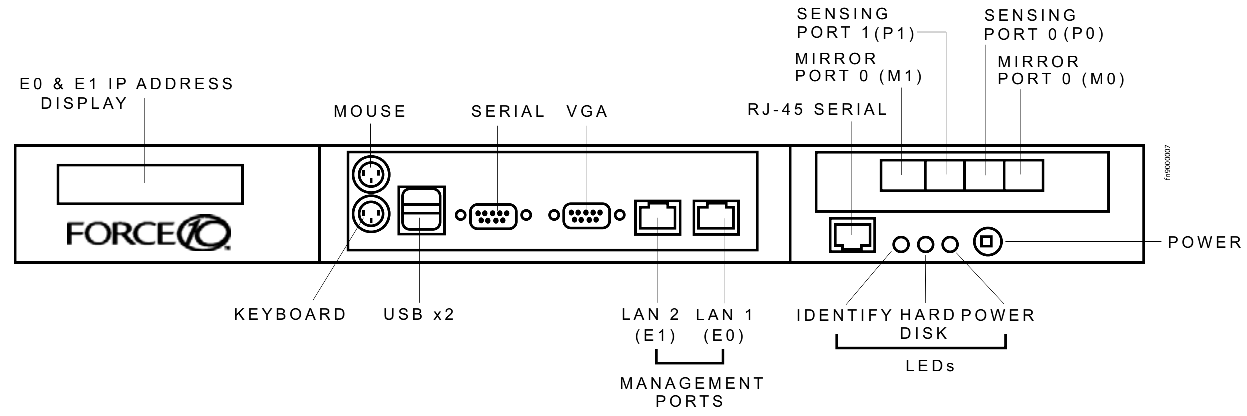


Figure 1 P-Series P1 Appliance (Front View)

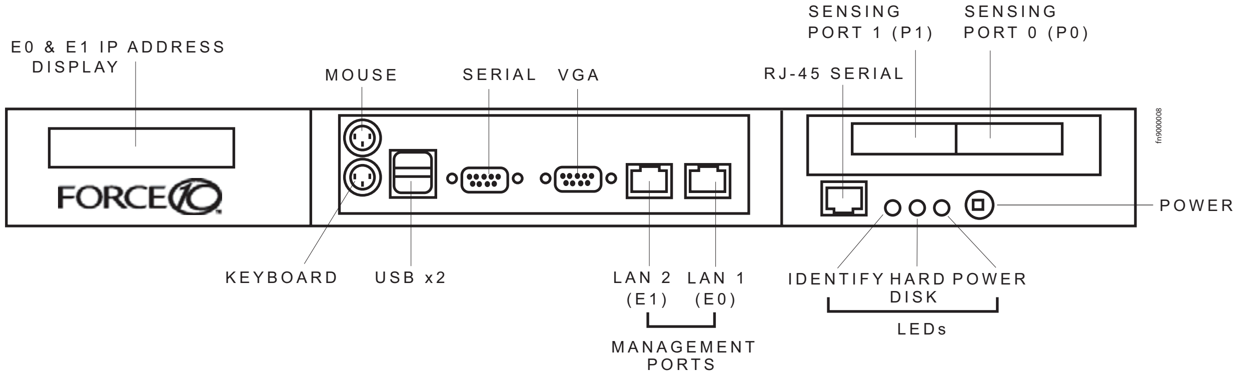


Figure 2 P-Series P10 Appliance (Front View)

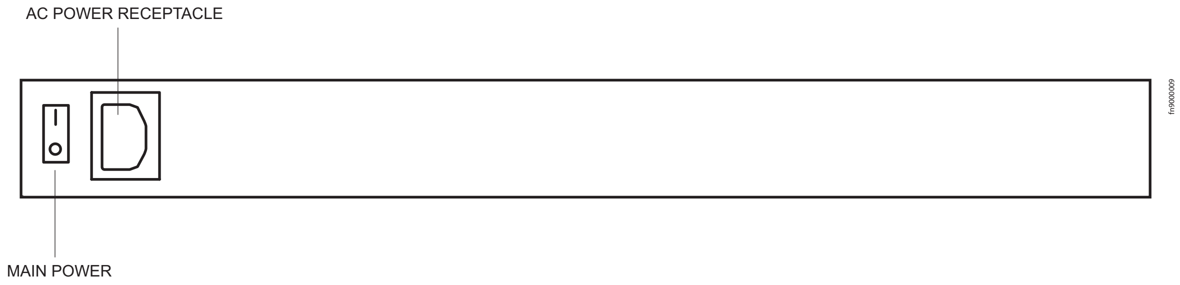


Figure 3 P-Series P1 & P10 Appliance (Rear View)

Label	Description
(LCD screen)	The LCD screen displays the IP address of the appliance next to either “e0:” or “e1:”, which represent LAN ports 1 and 2, respectively.
M1, M0 (on the P1)	These two ports are mirror ports through which copied matched traffic is routed. They accept the 1G small form-factor pluggable modules (SFP).
P1, P0 (on the P1)	These two ports are sensing ports through which traffic is routed. They accept the 1G small form-factor pluggable modules (SFP).
Port 1, Port 0 (on the P10)	These two ports are sensing ports through which traffic is routed. They accept 10G XPAK modules.
(unlabeled RJ-45 serial port next to IDENTIFY)	This port is not used.
IDENTIFY	This LED is not used.
HDD	This LED is blue when the hard disk is accessed.
PWR	This LED is green when the power is on.
(Power Button)	This button turns the appliance on and off. Press and hold the button to turn off the appliance.
(Laser Warning) CLASS 1 LASER PRODUCT LASERPRODUKT DER KLASSE 1 FN00048A	This label in the bottom right corner of the appliance indicates that the appliance is a Class 1 laser product that emits invisible laser radiation. This product complies with CDRH 21 CFR 1040.

Physical Connections

Note: Connections to the sensing, mirroring, and management ports require straight-through CAT5 cables.

Step	Task
1	Connect the power cable, a keyboard, and a monitor to the appliance.
2	Connect the LAN 1 port on the appliance to the local area network where DHCP is available. If a DHCP server is not available, an IP address can be assigned manually; see “Configuration” on page 13.
3	Install SFPs or XPAKs in the ports that will be used.
4	Connect the sensing ports to the devices from which the appliance will receive traffic. • Traffic originating from the device connected to Port 0 has Channel 0’s rules applied to it. • Traffic originating from the device connected to Port 1 has Channel 1’s rules applied to it.
5	Connect the mirroring ports to the devices that will receive mirrored traffic. • Mirror Port 0 mirrors matched traffic from Channel 0. • Mirror Port 1 mirrors matched traffic from Channel 1.

Step	Task
6	Connect the power cable to a power source, and switch on the main power on the back of the appliance.
7	Press the power button on the front of the appliance to turn on the device.

Booting

During booting, you are prompted to select one of two operating systems: Linux or FreeBSD.

- The Linux option represents Fedora Core 3.
- The FreeBSD option represents FreeBSD version 6.0.

If no selection is made, after a few seconds the Linux option is selected automatically.

The management ports are configured for DHCP and probe for an IP address, gateway, and name server. The IP address is displayed on the LCD screen.

When the appliance is powered up, all packets are forwarded between its ports by default until the firmware and device drivers are loaded. Once they have been loaded, the DPI generates interrupts to the host processor and offers the captured packets in the same way as a standard network interface card in promiscuous mode.

Configuration

Once the appliance is booted, perform the following configuration steps.

Step	Task
1	Log in as root with the password **plogin**.
2	Change the password, if desired, with the command **passwd**.
3	If DHCP is not available, use the **system-config-network** command to assign an IP address manually.
4	If desired, modify the host name or the network configuration using the command **system-config-network**. **Note:** Other **system-config-*** scripts are available to configure most other system options.

Security Check

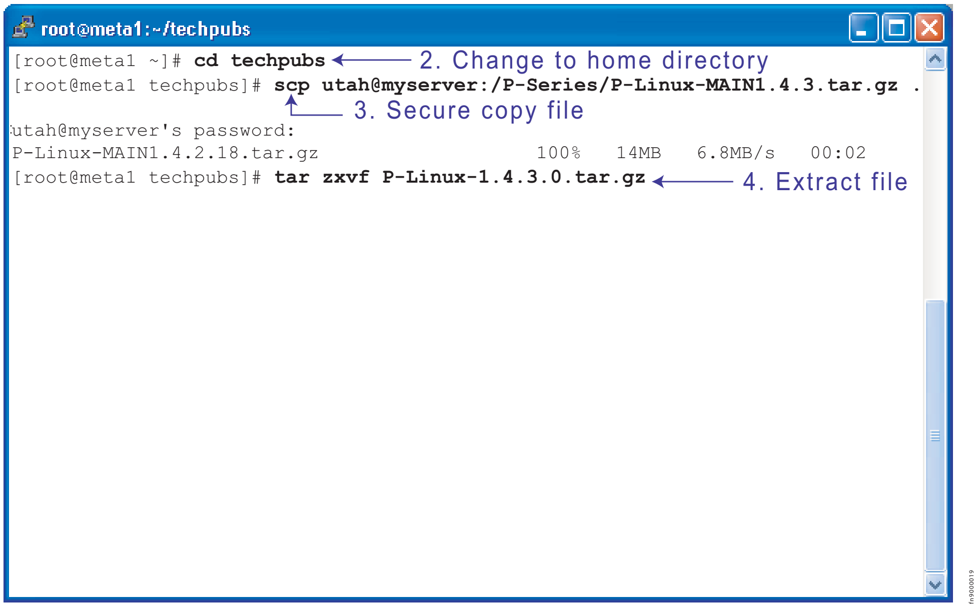
The P-Series appliances are remotely accessible only via Secure Shell Daemon (SSHD). However, inspect the configuration, and make sure it meets the security policy requirements of your network before deploying the appliance.

Upgrading to Version 1.4.3 or Higher

Note: You must be logged in as root to upgrade software.
Note: All of the figures in this section use Fedora Core 3 as the example OS.

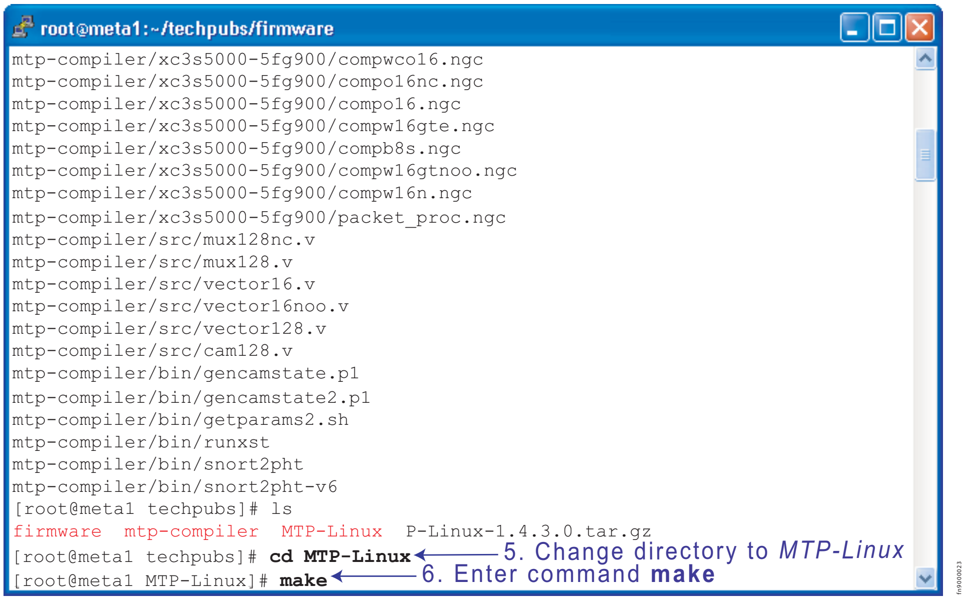
Step	Task	Command
1	Save earlier configuration files, if desired, by copying the files in the */usr/local/meta/0* directory to a temporary directory. **Note:** Firmware and configuration files from versions prior to version 1.4.3 cannot be reused.	**cp /usr/local/meta/0** *temporary directory*
2	• If you have not installed a previous version of the P-Series software, access the folder where you want to install it (*home directory*). • If you have installed a previous version of the P-Series software, access its folder and enter the **make erase** command to remove the software.	**cd** *home directory* **make erase**
3	Secure copy the file *P-Linux-1.4.3.{X}.tar.gz* or *P-FreeBSD-1.4.3.{X}.tar.gz* (based on the operating system you chose during booting) from a server to your home directory.	**scp** *username@server*:*absolute path/file* **.**
4	Extract the file (see Figure 4). • This command creates three sub-directories: *MTP-{Linux|FBSD-6.0}*, *firmware*, and *mtp-compiler* (see Figure 5).	**tar zxvf** *file*
5	Change directory to *MTP-{Linux|FBSD-6.0}* depending on which operating system you are using (see Figure 5).	**cd** *directory*
6	Enter the command **make** (see Figure 5). • The display appears as in Figure 6 when the make process is complete.	**make**
7	Enter the command **make install** (see Figure 6).	**make install**
8	Change directory to *firmware* (see Figure 7).	**cd ../firmware**

Step	Task	Command
9	Enter the command **make install** to install firmware (see Figure 7). • The display appears as in Figure 8 when the installation is complete.	**make install**



```
[root@meta1 ~]# cd techpubs
[root@meta1 techpubs]# scp utah@myserver:/P-Series/P-Linux-MAIN1.4.3.tar.gz .

utah@myserver's password:
P-Linux-MAIN1.4.2.18.tar.gz                    100%    14MB    6.8MB/s    00:02
[root@meta1 techpubs]# tar zxvf P-Linux-1.4.3.0.tar.gz
```

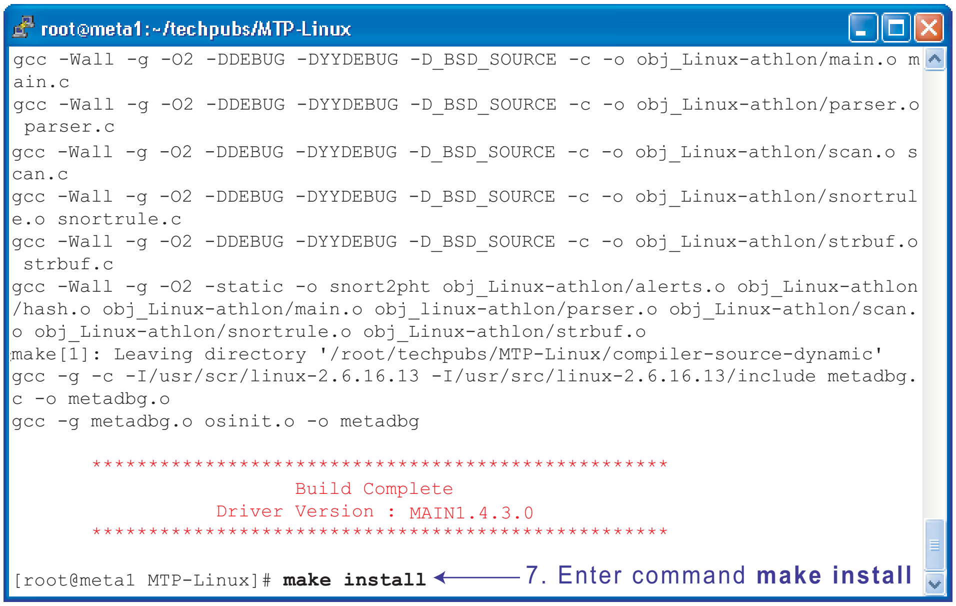
Figure 4 Software Installation Step 2-4



```
mtp-compiler/xc3s5000-5fg900/compwco16.ngc
mtp-compiler/xc3s5000-5fg900/compo16nc.ngc
mtp-compiler/xc3s5000-5fg900/compo16.ngc
mtp-compiler/xc3s5000-5fg900/compw16gte.ngc
mtp-compiler/xc3s5000-5fg900/compb8s.ngc
mtp-compiler/xc3s5000-5fg900/compw16gtnoo.ngc
mtp-compiler/xc3s5000-5fg900/compw16n.ngc
mtp-compiler/xc3s5000-5fg900/packet_proc.ngc
mtp-compiler/src/mux128nc.v
mtp-compiler/src/mux128.v
mtp-compiler/src/vector16.v
mtp-compiler/src/vector16noo.v
mtp-compiler/src/vector128.v
mtp-compiler/src/cam128.v
mtp-compiler/bin/gencamstate.p1
mtp-compiler/bin/gencamstate2.p1
mtp-compiler/bin/getparams2.sh
mtp-compiler/bin/runxst
mtp-compiler/bin/snort2pht
mtp-compiler/bin/snort2pht-v6
[root@meta1 techpubs]# ls
firmware  mtp-compiler  MTP-Linux  P-Linux-1.4.3.0.tar.gz
[root@meta1 techpubs]# cd MTP-Linux
[root@meta1 MTP-Linux]# make
```

root@meta1:~/techpubs/firmware
5. Change directory to MTP-Linux
6. Enter command make

Figure 5 Software Installation Step 5-6

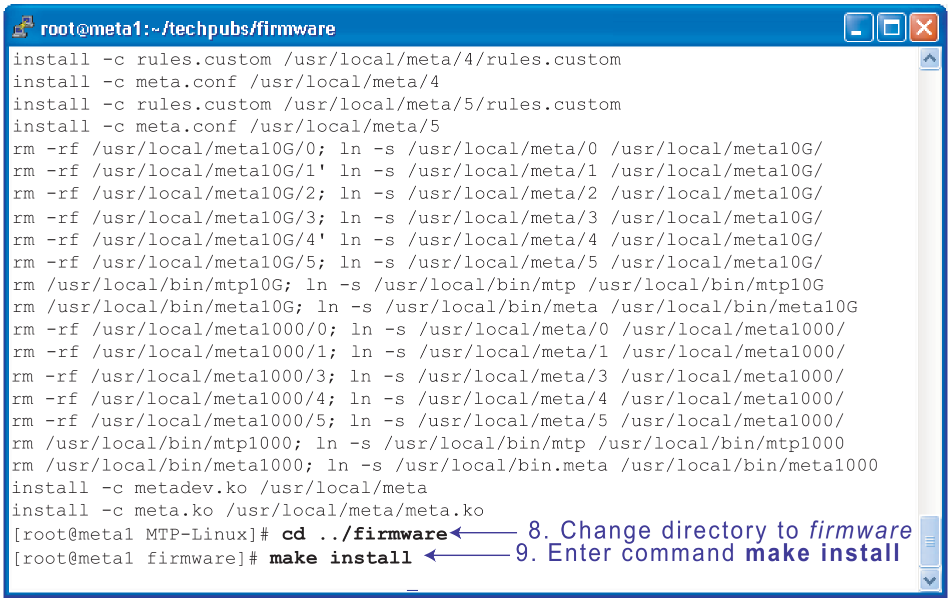


root@meta1:~/techpubs/MTP-Linux

```
gcc -Wall -g -O2 -DDEBUG -DYYDEBUG -D_BSD_SOURCE -c -o obj_Linux-athlon/main.o m
ain.c
gcc -Wall -g -O2 -DDEBUG -DYYDEBUG -D_BSD_SOURCE -c -o obj_Linux-athlon/parser.o
 parser.c
gcc -Wall -g -O2 -DDEBUG -DYYDEBUG -D_BSD_SOURCE -c -o obj_Linux-athlon/scan.o s
can.c
gcc -Wall -g -O2 -DDEBUG -DYYDEBUG -D_BSD_SOURCE -c -o obj_Linux-athlon/snortrul
e.o snortrule.c
gcc -Wall -g -O2 -DDEBUG -DYYDEBUG -D_BSD_SOURCE -c -o obj_Linux-athlon/strbuf.o
 strbuf.c
gcc -Wall -g -O2 -static -o snort2pht obj_Linux-athlon/alerts.o obj_Linux-athlon
/hash.o obj_Linux-athlon/main.o obj_linux-athlon/parser.o obj_Linux-athlon/scan.
o obj_Linux-athlon/snortrule.o obj_Linux-athlon/strbuf.o
make[1]: Leaving directory '/root/techpubs/MTP-Linux/compiler-source-dynamic'
gcc -g -c -I/usr/scr/linux-2.6.16.13 -I/usr/src/linux-2.6.16.13/include metadbg.
c -o metadbg.o
gcc -g metadbg.o osinit.o -o metadbg

       ***************************************************
                         Build Complete
                  Driver Version : MAIN1.4.3.0
       ***************************************************

[root@meta1 MTP-Linux]# make install
```

7. Enter command **make install**

Figure 6 Software Installation Step 7



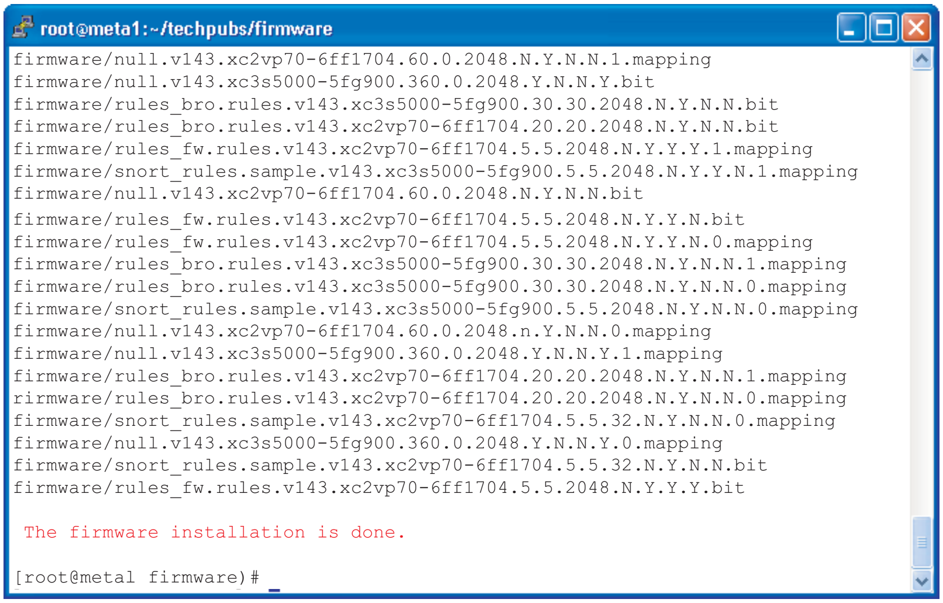
root@meta1:~/techpubs/firmware

```
install -c rules.custom /usr/local/meta/4/rules.custom
install -c meta.conf /usr/local/meta/4
install -c rules.custom /usr/local/meta/5/rules.custom
install -c meta.conf /usr/local/meta/5
rm -rf /usr/local/meta10G/0; ln -s /usr/local/meta/0 /usr/local/meta10G/
rm -rf /usr/local/meta10G/1' ln -s /usr/local/meta/1 /usr/local/meta10G/
rm -rf /usr/local/meta10G/2; ln -s /usr/local/meta/2 /usr/local/meta10G/
rm -rf /usr/local/meta10G/3; ln -s /usr/local/meta/3 /usr/local/meta10G/
rm -rf /usr/local/meta10G/4' ln -s /usr/local/meta/4 /usr/local/meta10G/
rm -rf /usr/local/meta10G/5; ln -s /usr/local/meta/5 /usr/local/meta10G/
rm /usr/local/bin/mtp10G; ln -s /usr/local/bin/mtp /usr/local/bin/mtp10G
rm /usr/local/bin/meta10G; ln -s /usr/local/bin/meta /usr/local/bin/meta10G
rm -rf /usr/local/meta1000/0; ln -s /usr/local/meta/0 /usr/local/meta1000/
rm -rf /usr/local/meta1000/1; ln -s /usr/local/meta/1 /usr/local/meta1000/
rm -rf /usr/local/meta1000/3; ln -s /usr/local/meta/3 /usr/local/meta1000/
rm -rf /usr/local/meta1000/4; ln -s /usr/local/meta/4 /usr/local/meta1000/
rm -rf /usr/local/meta1000/5; ln -s /usr/local/meta/5 /usr/local/meta1000/
rm /usr/local/bin/mtp1000; ln -s /usr/local/bin/mtp /usr/local/bin/mtp1000
rm /usr/local/bin/meta1000; ln -s /usr/local/bin.meta /usr/local/bin/meta1000
install -c metadev.ko /usr/local/meta
install -c meta.ko /usr/local/meta/meta.ko
[root@meta1 MTP-Linux]# cd ../firmware
[root@meta1 firmware]# make install
```

8. Change directory to *firmware*

9. Enter command **make install**

Figure 7 Software Installation Step 8-9



```
firmware/null.v143.xc2vp70-6ff1704.60.0.2048.N.Y.N.N.1.mapping
firmware/null.v143.xc3s5000-5fg900.360.0.2048.Y.N.N.Y.bit
firmware/rules_bro.rules.v143.xc3s5000-5fg900.30.30.2048.N.Y.N.N.bit
firmware/rules_bro.rules.v143.xc2vp70-6ff1704.20.20.2048.N.Y.N.N.bit
firmware/rules_fw.rules.v143.xc2vp70-6ff1704.5.5.2048.N.Y.Y.Y.1.mapping
firmware/snort_rules.sample.v143.xc3s5000-5fg900.5.5.2048.N.Y.Y.N.1.mapping
firmware/null.v143.xc2vp70-6ff1704.60.0.2048.N.Y.N.N.bit
firmware/rules_fw.rules.v143.xc2vp70-6ff1704.5.5.2048.N.Y.Y.N.bit
firmware/rules_fw.rules.v143.xc2vp70-6ff1704.5.5.2048.N.Y.Y.N.0.mapping
firmware/rules_bro.rules.v143.xc3s5000-5fg900.30.30.2048.N.Y.N.N.1.mapping
firmware/rules_bro.rules.v143.xc3s5000-5fg900.30.30.2048.N.Y.N.N.0.mapping
firmware/snort_rules.sample.v143.xc3s5000-5fg900.5.5.2048.N.Y.N.N.0.mapping
firmware/null.v143.xc2vp70-6ff1704.60.0.2048.n.Y.N.N.0.mapping
firmware/null.v143.xc3s5000-5fg900.360.0.2048.Y.N.N.Y.1.mapping
firmware/rules_bro.rules.v143.xc2vp70-6ff1704.20.20.2048.N.Y.N.N.1.mapping
rirmware/rules_bro.rules.v143.xc2vp70-6ff1704.20.20.2048.N.Y.N.N.0.mapping
firmware/snort_rules.sample.v143.xc2vp70-6ff1704.5.5.32.N.Y.N.N.0.mapping
firmware/null.v143.xc3s5000-5fg900.360.0.2048.Y.N.N.Y.0.mapping
firmware/snort_rules.sample.v143.xc2vp70-6ff1704.5.5.32.N.Y.N.N.bit
firmware/rules_fw.rules.v143.xc2vp70-6ff1704.5.5.2048.N.Y.Y.Y.bit

 The firmware installation is done.

[root@metal firmware)# _
```

Figure 8 Software Installation Complete

Chapter 2 Getting Started

To begin inspecting and filtering traffic you must:

1. Select firmware and dynamic rules
2. Set capture/forward policies
3. Check for proper operation by generating traffic across the appliance

Step	Task
1	As root, enter the command **mtp** from the Unix command line to invoke a graphical user interface (GUI).
2	Enter the command **m** from the GUI command line.
3	Select **Manage Firmware** from the Rule Management GUI, and select “null” firmware for the appliance you are using. • The firmware *null.null.v143.xc3s5000-4fg900-2* is for the P1. • The firmware *null.null.v143.xc2vp70-6ff1704* is for the P10. **Note:** The sample firmware and rules files are testing examples only. Force 10 recommends not employing the sample firmware for production IDS/IPS use.
4	Select **Done** and confirm your selection.
5	Select **Edit Rules** from the Rule Management GUI.
6	Uncomment the rule **alert on c0 icmp any any -> any any (msg:"@icmp";)** by removing the **#** symbol before the rule. • Enter the command **i** to enter insert mode. • Navigate to the character using the arrow keys, and delete the character.
7	Enter the command **:q!** to exit the *vi* editor, and confirm your changes.
8	Select **Manage Rules** from the Rule Management GUI.
9	Select the capture/forward policy *divert* for the rule: **alert on c0 icmp any any -> any any (msg:"@icmp";)**.
10	Select **Done,** and confirm your selections.
11	Select **Exit** from the Rule Management GUI.
12	Run a packet sniffer such as *tcpdump* on the network interface associated with the appliance.

Step	Task
13	Generate some ICMP traffic to be exchanged between endpoints. • *Endpoints* are two network nodes on opposite sides of the appliance such that traffic between those nodes passes through the appliance. • For example, enter **ping** *destaddress*, where *destaddress* is the IP address of the endpoint on the opposite end of the appliance.
14	If you are using *tcpdump*, enter the command **tcpdump -i eth2 -n** from the Unix command line. • This prints to standard output all of the packets captured by the DPI (assuming the appliance registers as *eth2*). • If the appliance is operating correctly, you will see the ICMP packets after a few seconds, when the receive buffer is full.

Chapter 3 Introduction

The P-Series P1 and P10 *Intrusion Detection and Prevention System* (*IDS/IPS*) appliances are driven by a proprietary hardware-based engine called *Dynamic Parallel Inspection* (*DPI*).

DPI includes a Multiple Instruction Single Data (MISD) massively parallel processor that can execute thousands of security policies or traffic capture operations on the same data stream at the same time.

DPI synthesizes individual security policies and packet analysis algorithms and maps them directly into silicon hardware "gates." Through this design it is able to deliver full packet inspection and protection at line rate for 1-Gigabit and 10-Gigabit links whether the traffic load or security policy is 1% or 100%.

The policies can be derived from public domain signatures, or they can be completely user-defined. For each policy, you can direct the DPI to:

- Capture packets for the host
- Forward packets (with negligible delay)
- Block packets (by corrupting the CRC)

As a result, the P-Series appliances can be used as both IDS accelerators and stateful content filters for IPS applications. In an active configuration, they can be inserted inline into the network; this alleviates the need for a SPAN port or tap and enables filtering applications. In passive configurations, the appliances can merely listen to the network via a mirroring port or tap.

Hardware Architecture Overview

The P1 and P10 are 1 RU appliances provisioned with AMD Dual Opteron 280 processors, a 400-GB hard drive, 2 GB of RAM, and one DPI processing system.

The DPI system is provisioned with two FPGAs; one is dedicated to ensuring line rate traffic forwarding; the other is dedicated to DPI performance. In addition, there is SRAM state table memory for 2 million flows on the P1 and 8 million flows on the P10.

Figure 9 shows packet flow in the DPI of the P10, which is a two-port device. Packets are forwarded from the receive side of the first port (Rx0) to the transmit side of the second port (Tx1). Likewise, Rx1 forwards packets to Tx0 of the first port.

As the packets are being forwarded they are also processed in real time by two independent processing channels, each with its own set of policies. If there is a match in a processing channel, the DPI can corrupt the packet’s CRC, capture it, and send it to the host through the PCI bus. The two processing channels are completely independent, and thus they can be used to process two asymmetric links, or both directions of a full duplex connection.

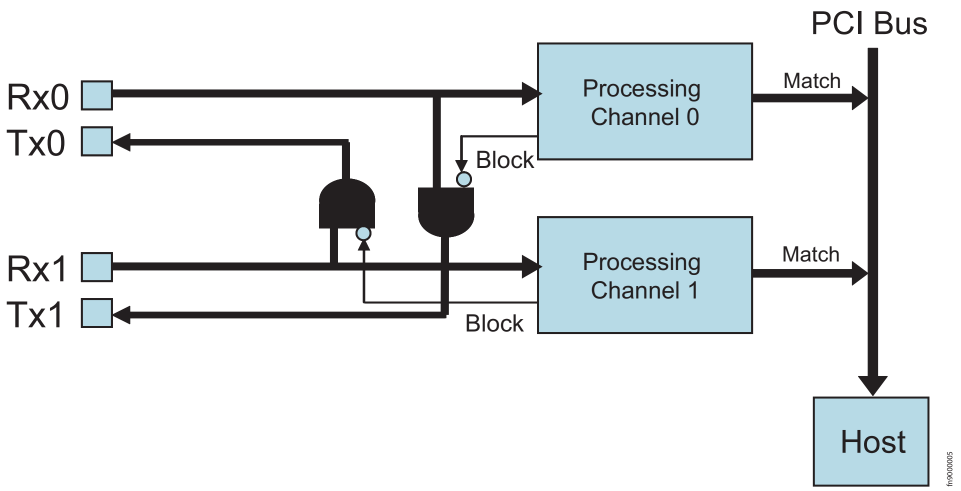


Figure 9 Logic Diagram of Traffic Flow in the P10 DPI

Mirroring in the P1

In addition to two sensing interfaces, the P1 includes two 1-Gigabit Ethernet mirroring ports. These ports can copy and forward matched traffic to another device. It is also possible to disable the PCI bus capture, and let the fast-path bypass the host entirely for applications in which host capture is not desired.

Figure 10 illustrates how all matched packets are copied and transmitted by mirror ports.

Note: The P10 appliance does not have mirroring capabilities.

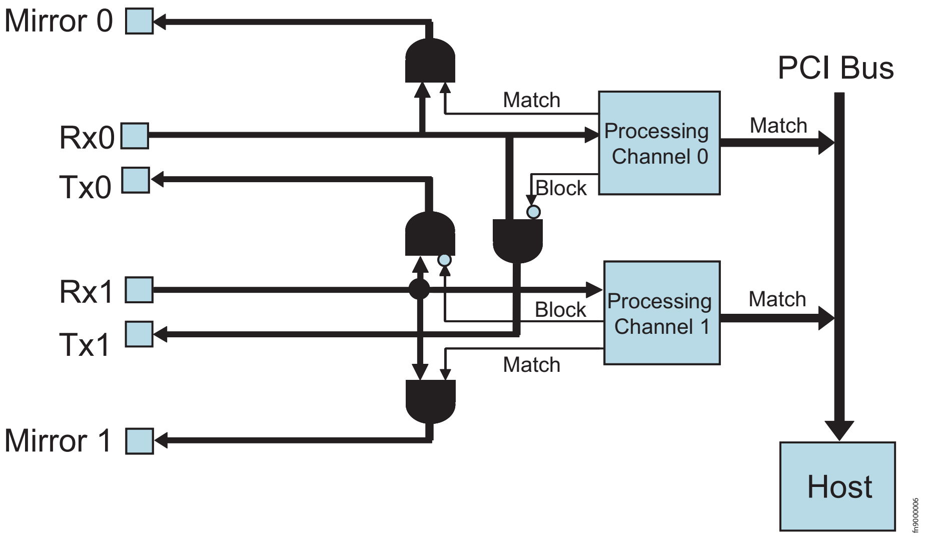


Figure 10 Logic Diagram of Traffic Flow and Mirroring in the P1 DPI

Types of Rules

Two types of rules can be uploaded to the FPGA:

- *Static rules*: Static rules are compiled to become part of the firmware and are mapped directly into logic gates. Static rules can be disabled/enabled individually, but they cannot be changed once they have been loaded into the FPGA.
- *Dynamic rules:* Dynamic rules are programmed at runtime in the DPI hardware registers and can be configured without changing the firmware. These rules (like static rules) can be disabled/enabled individually.

Sample Rules and Firmware

The P-Series appliances includes sample rules files in the *mtp-compiler/rules* directory. You can browse these files in order to become more familiar with Snort syntax or creating rules files; you can also generate firmware from these files at your discretion.

Firmware is a set of rules that has been transformed—using a compiler—from Snort syntax into a form suitable for uploading to the FPGA. Four sets of sample rules files have been compiled into firmware and are available to be uploaded to the FPGA using either of two firmware management methods (see "Rule Management" on page 24). Table 1 describes each sample rules file.

Table 1 Sample Rules Files

Rule Set	Description
snort.rules	This file contains rules written in Snort syntax that were derived from publicly available IDS rules.
null.rules	This file contains no rules; the firmware created from these files are empty images that maximize the dynamic rule capacity (see "Rules Capacity" on page 39).
fw.rules	This file is not used.
bro.rules	This file contains rules that could be used to inter-operate with the Bro IDS system.

The firmware based on the sample rules files will follow the naming convention described in "Selecting Firmware with the GUI" on page 31.

Note: Force 10 recommends not using the sample firmware for production IDS/IPS use. The sample firmware requires considerable site-specific customization in order to be effective; they are included only for you to become more familiar with the functionality of the appliance.

Rule Management

The P-Series software provides two methods by which you can manage the rules and functionality of the appliance:

- *Graphical User Interface*: The graphical user interface (GUI) is a menu-based method for managing the appliance.
- *Command Line Interface*: The command line interface (CLI) uses a script called *meta* through which you can manually perform the same management tasks as the GUI by entering commands at the command prompt.

You can use both the GUI and CLI to manage the appliance remotely.

Force10 recommends using the GUI if no programmatic interface is required.

Chapter 4 Graphical User Interface

The GUI can be used to:

- Start and stop the DPI
- Load firmware
- Compile and load dynamic rules
- Manage the runtime parameters
- Manage the capture/forward policies for rules

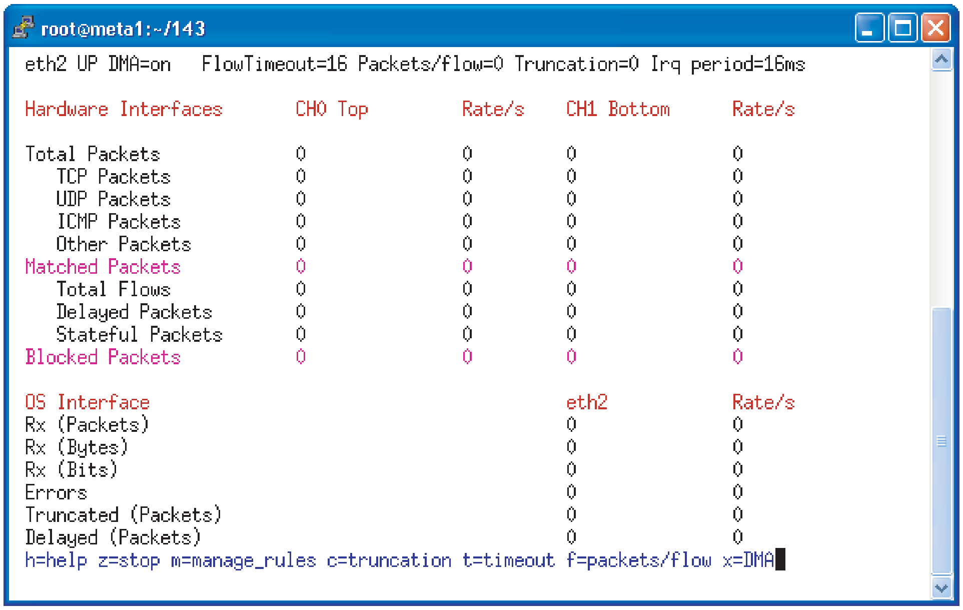
Note: The GUI requires the super user privilege.

Step	Task
1	Invoke the GUI by entering the command **mtp**. • The OS environment variables are set such that the **mtp** command can be executed from any path.

Runtime statistics are displayed after the **mtp** command is executed. If the FPGA is not loaded, the display does not show any runtime statistics except for the CPU usage, and it appears as shown in Figure 11. If firmware is loaded, the display will appear as in Figure 17.

GUI Commands

From the Runtime Statistics display, you can enter commands to control the DPI (see Table 2, or enter the **h** command from the GUI command line).



```
eth2 UP DMA=on   FlowTimeout=16 Packets/flow=0 Truncation=0 Irq period=16ms

Hardware Interfaces        CH0 Top          Rate/s     CH1 Bottom       Rate/s

Total Packets              0                0          0                0
   TCP Packets             0                0          0                0
   UDP Packets             0                0          0                0
   ICMP Packets            0                0          0                0
   Other Packets           0                0          0                0
Matched Packets            0                0          0                0
   Total Flows             0                0          0                0
   Delayed Packets         0                0          0                0
   Stateful Packets        0                0          0                0
Blocked Packets            0                0          0                0

OS Interface                                           eth2             Rate/s
Rx (Packets)                                           0                0
Rx (Bytes)                                             0                0
Rx (Bits)                                              0                0
Errors                                                 0                0
Truncated (Packets)                                    0                0
Delayed (Packets)                                      0                0
h=help z=stop m=manage_rules c=truncation t=timeout f=packets/flow x=DMA
```

Figure 11 Runtime Statistics - FPGA Unloaded

Note: Commands that require a subsequent value entry have the current value displayed in parentheses at the prompt.

Table 2 GUI Commands

Command	Description
a	This command establishes the IRQ period (measured in milliseconds), which moderates DPI access to the PCI bus. A value of *0* allows the DPI to gain full access to the bus and is recommended.
c	This command establishes the number of bytes to capture after a match (Truncation). A value of *0* specifies the entire packet.
d	This command brings the OS network interface down and disables matching.
f	This command establishes the maximum number of packets to be captured for each flow (Packets/ Flow). A value of *0* specifies all packets.
h	This command displays help information about the commands.
i	This command establishes the display refresh interval (measured in seconds).
m	This command invokes a dialog menu through which dynamic rules can be defined and capture/ forwarding policies can be set for each individual rule (see Figure 12).
q	This command exits the graphical user interface.
s	This command starts or restarts the drivers and reloads the firmware.
t	This command establishes the number of seconds after which a flow is considered expired (Flow Timeout).

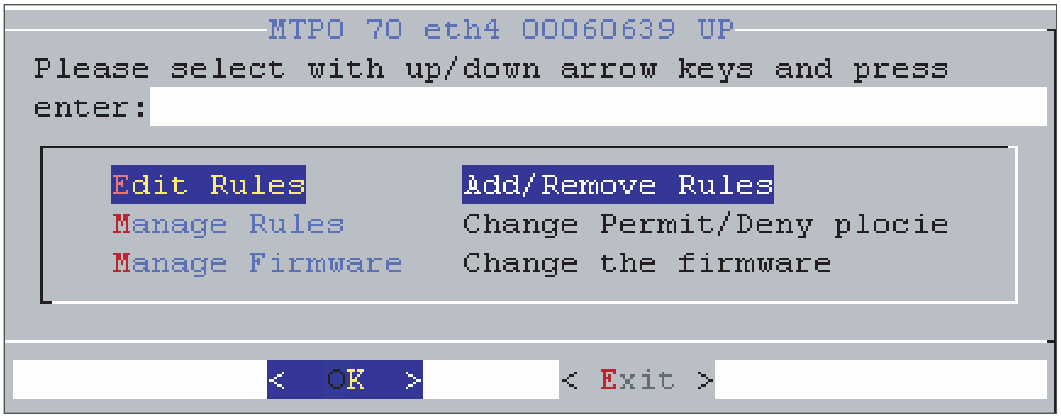
Table 2 GUI Commands

Command	Description
u	This command brings the OS network interface up and enables matching. This is similar to the command **s**, but it does not load/reload the driver. It is only valid after the command **s** has been executed.
x	This command toggles the direct memory access (DMA) off and on to enable or disable capturing to the host, respectively.

Note: Commands **1**, **2**, **3**, **4**, and **5** are for engineering use only. If you enter a command **1** through **5** by mistake, enter **0** to return to the runtime statistics screen.

Managing Rules, Policies, and Firmware

Enter the **m** command from the GUI command line (see “GUI Commands” on page 25) to invoke a menu that enables you to manage dynamic rules, capture/forward policies, and firmware. Three options are available; they are shown in Figure 12 and described in Table 3.



```
MTPO 70 eth4 00060639 UP
Please select with up/down arrow keys and press
enter:
```

Figure 12 Rule Management GUI

Table 3 Managing Rules Using the GUI

Option	Description
Edit Rules	This option invokes the *vi* editor on the file *rules.custom* in the */user/local/meta/0* directory (see “Editing Dynamic Rules with the GUI” on page 29). • You can add, delete, or modify dynamic rules for either of the processing channels (see Appendix A , on page 59 for information on *vi*). • The rules are automatically compiled and loaded into the appliance; you are prompted to confirm these actions. **Note:** Dynamic rules only apply to IPv4 traffic. IPv6 traffic should be monitored or controlled through static rules.
Manage Rules	This option instructs the DPI on handling matching packets. • It displays a list of all the rules contained in the FPGA and the policy setting for each. • There are four policies available and they are described in Table 4. • Rules configured to ignore a packet—that is, the policy setting is *permit* or *deny*—take precedence over rules that have a policy setting of *alert* or *divert*. Therefore, a *permit* or *deny* rule will disable the capturing for all other rules that match the same packet. • To modify policy settings, see “Managing Capture/Forward Policies with the GUI” on page 30. **Note:** The **Capture** toggle is not used. **Note:** Capture/forward settings can only be modified through the graphical user interface.
Manage Firmware	It displays the firmware files in */usr/local/meta/firmware* and allows you to select one to be uploaded to the FPGA. Selecting firmware restarts and reloads the FPGA. To manage firmware, see “Selecting Firmware with the GUI” on page 31.

Table 4 describes the four possible combinations of capture/forward policies.

Table 4 Capture/Forward Policies

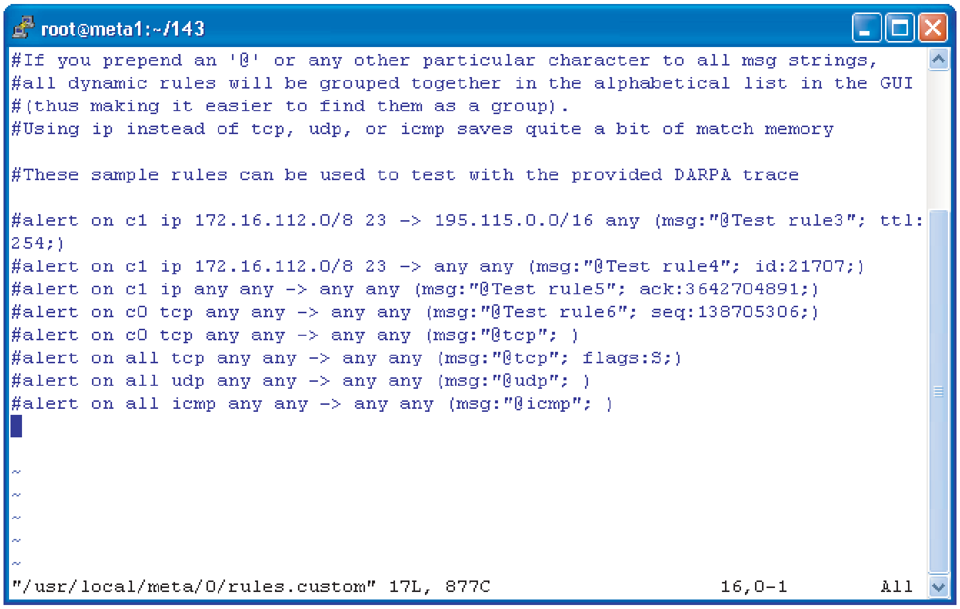
Policy	Capture	Forward
Permit		X
Deny		
Alert	X	X
Divert	X	

Editing Dynamic Rules with the GUI

Dynamic rules are stored in the file *rules.custom* in the */usr/local/meta/0* directory. The GUI provides a quick way to access and modify these rules by invoking the *vi* editor on this file.

Follow these steps to modify dynamic rules:

Step	Task
1	Enter the **m** command from the GUI command line (see “GUI Commands” on page 25) to access the main rule management GUI (see Figure 12).
2	Select **Edit Rules** to invoke the *vi* editor (see Figure 13).
3	Add, delete, alter, or uncomment rules using *vi* commands (see Appendix A , on page 59).
4	You will be prompted to confirm your changes upon exiting the editor.



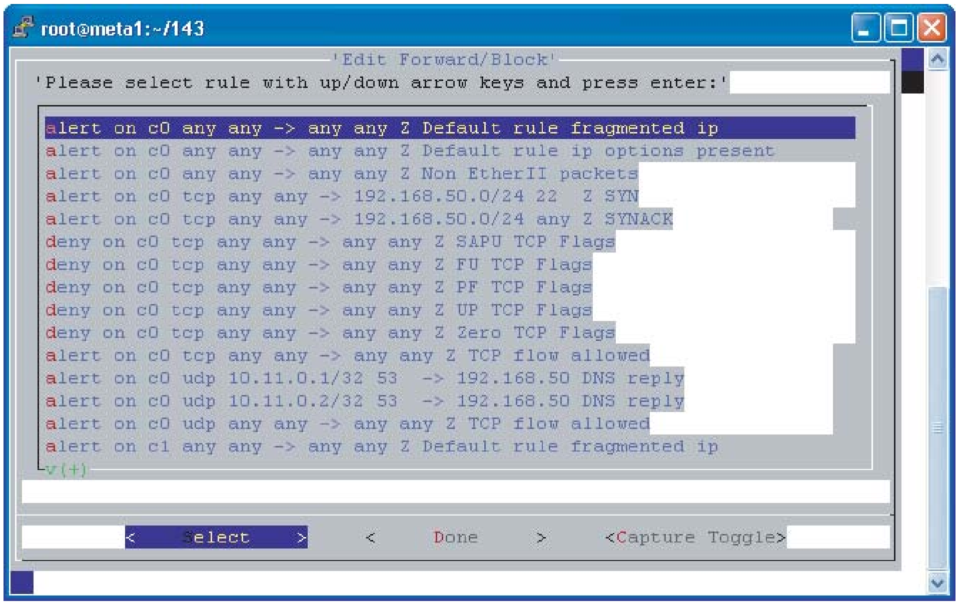
```
#If you prepend an '@' or any other particular character to all msg strings,
#all dynamic rules will be grouped together in the alphabetical list in the GUI
#(thus making it easier to find them as a group).
#Using ip instead of tcp, udp, or icmp saves quite a bit of match memory

#These sample rules can be used to test with the provided DARPA trace

#alert on c1 ip 172.16.112.0/8 23 -> 195.115.0.0/16 any (msg:"@Test rule3"; ttl:
254;)
#alert on c1 ip 172.16.112.0/8 23 -> any any (msg:"@Test rule4"; id:21707;)
#alert on c1 ip any any -> any any (msg:"@Test rule5"; ack:3642704891;)
#alert on c0 tcp any any -> any any (msg:"@Test rule6"; seq:138705306;)
#alert on c0 tcp any any -> any any (msg:"@tcp"; )
#alert on all tcp any any -> any any (msg:"@tcp"; flags:S;)
#alert on all udp any any -> any any (msg:"@udp"; )
#alert on all icmp any any -> any any (msg:"@icmp"; )


~
~
~
~
~
"/usr/local/meta/0/rules.custom" 17L, 877C                    16,0-1        All
```

Figure 13 Editing Dynamic Rules in *vi*

Managing Capture/Forward Policies with the GUI

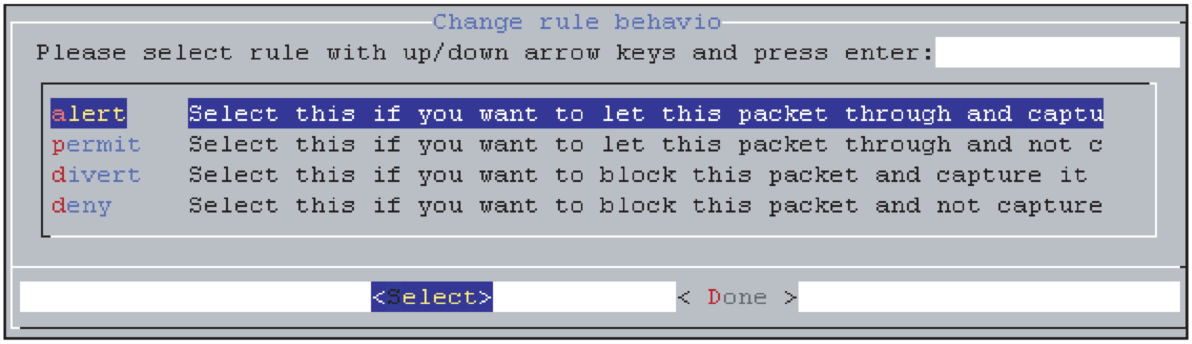
Upon compiling static and dynamic rules, default capture/forward policies are assigned to each rule. Follow these steps to change capture/forward policies:

Step	Task
1	Enter the **m** command from the GUI command line (see “GUI Commands” on page 25) to access the rule management GUI (see Figure 12).
2	Select **Manage Rules** to access the policy management menu (see Figure 14).
3	Use the arrow keys to highlight a rule and the **Select** option, and press the *Enter* key.
4	Select *alert*, *permit*, *divert* or *deny*, based on the descriptions in Table 4 (also see Figure 15).
5	Exit the menu by selecting **Done**, and repeat Steps 3 through 5 for other rules, if desired.
6	Select **Done**; you will be prompted to confirm your changes.



```
                         'Edit Forward/Block'
 'Please select rule with up/down arrow keys and press enter:'

   alert on c0 any any -> any any Z Default rule fragmented ip
   alert on c0 any any -> any any Z Default rule ip options present
   alert on c0 any any -> any any Z Non EtherII packets
   alert on c0 tcp any any -> 192.168.50.0/24 22  Z SYN
   alert on c0 tcp any any -> 192.168.50.0/24 any Z SYNACK
   deny on c0 tcp any any -> any any Z SAPU TCP Flags
   deny on c0 tcp any any -> any any Z FU TCP Flags
   deny on c0 tcp any any -> any any Z PF TCP Flags
   deny on c0 tcp any any -> any any Z UP TCP Flags
   deny on c0 tcp any any -> any any Z Zero TCP Flags
   alert on c0 tcp any any -> any any Z TCP flow allowed
   alert on c0 udp 10.11.0.1/32 53  -> 192.168.50 DNS reply
   alert on c0 udp 10.11.0.2/32 53  -> 192.168.50 DNS reply
   alert on c0 udp any any -> any any Z TCP flow allowed
   alert on c1 any any -> any any Z Default rule fragmented ip
  v(+)

       <     Select     >     <     Done     >     <Capture Toggle>
```

Figure 14 Managing Capture/Forward Policies GUI



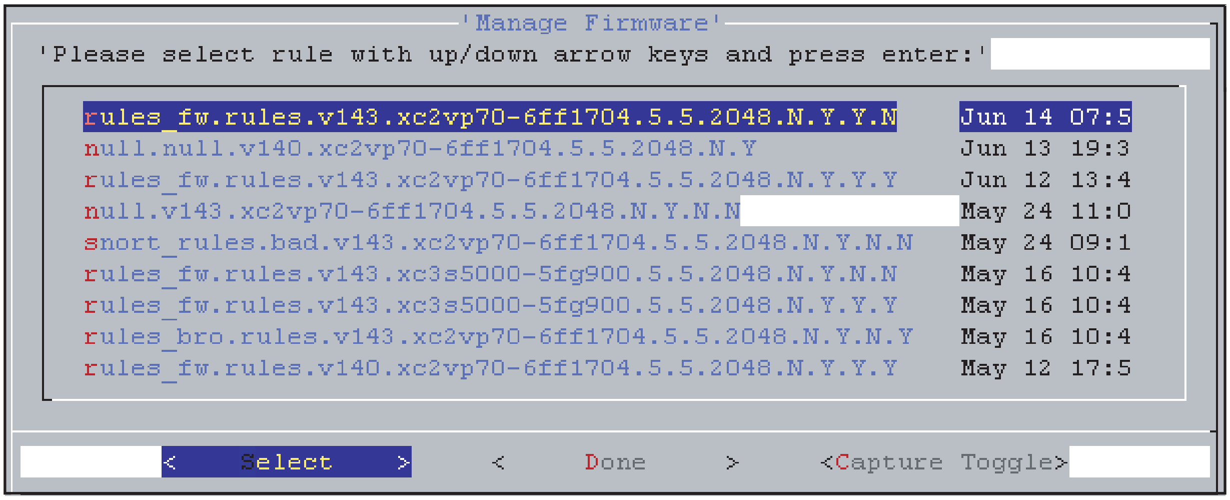
```
                         Change rule behavio
 Please select rule with up/down arrow keys and press enter:

  alert     Select this if you want to let this packet through and captu
  permit    Select this if you want to let this packet through and not c
  divert    Select this if you want to block this packet and capture it
  deny      Select this if you want to block this packet and not capture

                     <Select>              <  Done  >
```

Figure 15 Capture/Forward Policies GUI

Selecting Firmware with the GUI

Firmware is a set of rules that has been transformed—using a compiler—from Snort syntax into a form suitable for uploading to the FPGA. To select firmware:

Step	Task
1	Enter the **m** command from the GUI command line (see “GUI Commands” on page 25) to access the main rule management GUI.
2	Select **Manage Firmware** (see Figure 16).
3	Use the arrow keys to highlight the desired firmware and the **Select** option, and press the *Enter* key. See “Firmware Filenames” on page 46 for information on identifying firmware by their filenames.
4	Confirm your selection, and exit the GUI.



```
                          'Manage Firmware'
'Please select rule with up/down arrow keys and press enter:'

   rules_fw.rules.v143.xc2vp70-6ff1704.5.5.2048.N.Y.Y.N     Jun 14 07:5
   null.null.v140.xc2vp70-6ff1704.5.5.2048.N.Y              Jun 13 19:3
   rules_fw.rules.v143.xc2vp70-6ff1704.5.5.2048.N.Y.Y.Y     Jun 12 13:4
   null.v143.xc2vp70-6ff1704.5.5.2048.N.Y.N.N               May 24 11:0
   snort_rules.bad.v143.xc2vp70-6ff1704.5.5.2048.N.Y.N.N    May 24 09:1
   rules_fw.rules.v143.xc3s5000-5fg900.5.5.2048.N.Y.N.N     May 16 10:4
   rules_fw.rules.v143.xc3s5000-5fg900.5.5.2048.N.Y.Y.Y     May 16 10:4
   rules_bro.rules.v143.xc2vp70-6ff1704.5.5.2048.N.Y.N.Y    May 16 10:4
   rules_fw.rules.v140.xc2vp70-6ff1704.5.5.2048.N.Y.Y.Y     May 12 17:5

       <    Select    >      <     Done     >     <Capture Toggle>
```

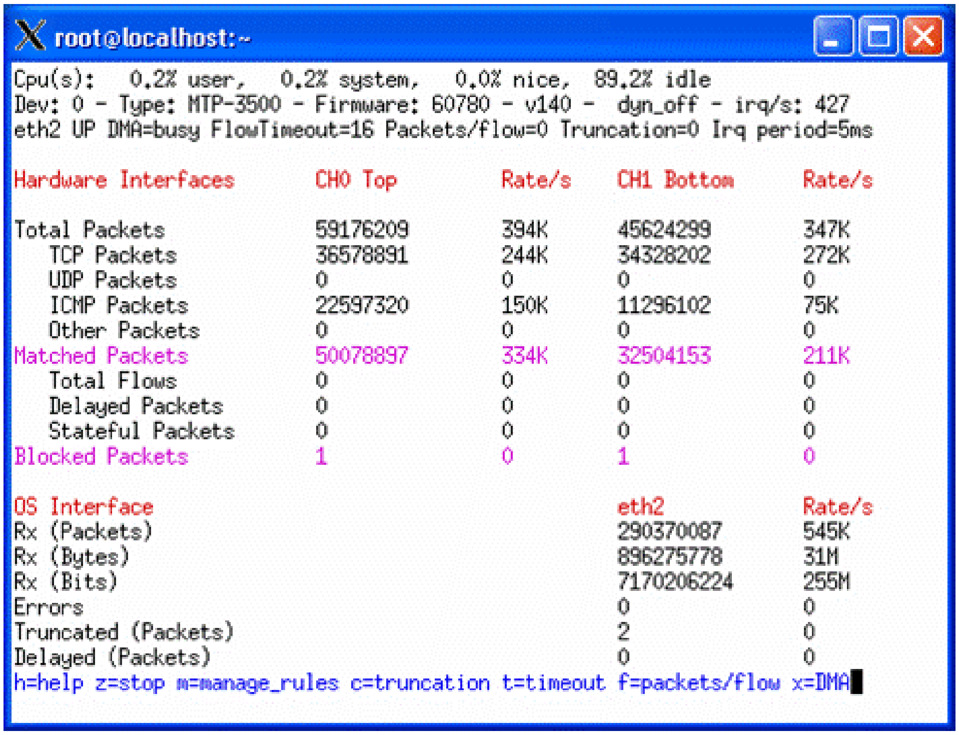
Figure 16 Manage Firmware GUI

Runtime Statistics

Runtime statistics are displayed when firmware is uploaded, and traffic is flowing across the appliance (see Figure 17).

- The first line of runtime statistics shows the cumulative CPU usage.
- The second line shows the device number, type of device, firmware ID, and interrupt rate generated by the device. The interrupt rate should be at most 2000/s and can be lowered by increasing the IRQ period.
- The third line shows the status of the Ethernet interface and direct memory access (DMA), and the values of Flow Timeout, Packets/Flow, Truncation, and IRQ Period. These parameters can be adjusted using the GUI commands described in Table 2.

The remaining lines report the cumulative number of events and the rate of those events. A description of each line is given in Table 5.



```
Cpu(s):   0.2% user,   0.2% system,   0.0% nice,  89.2% idle
Dev: 0 - Type: MTP-3500 - Firmware: 60780 - v140 -  dyn_off - irq/s: 427
eth2 UP DMA=busy FlowTimeout=16 Packets/flow=0 Truncation=0 Irq period=5ms

Hardware Interfaces         CH0 Top         Rate/s    CH1 Bottom      Rate/s

Total Packets               59176209        394K      45624299        347K
   TCP Packets              36578891        244K      34328202        272K
   UDP Packets              0               0         0               0
   ICMP Packets             22597320        150K      11296102        75K
   Other Packets            0               0         0               0
Matched Packets             50078897        334K      32504153        211K
   Total Flows              0               0         0               0
   Delayed Packets          0               0         0               0
   Stateful Packets         0               0         0               0
Blocked Packets             1               0         1               0

OS Interface                                          eth2            Rate/s
Rx (Packets)                                          290370087       545K
Rx (Bytes)                                            896275778       31M
Rx (Bits)                                             7170206224      255M
Errors                                                0               0
Truncated (Packets)                                   2               0
Delayed (Packets)                                     0               0
h=help z=stop m=manage_rules c=truncation t=timeout f=packets/flow x=DMA
```

Figure 17 Runtime Statistics - FPGA Loaded

Table 5 Runtime Statistics Description

Statistic	Description
Total Packets	This line shows the number of packets received by the ports. This is a Layer 1 statistic and is independent of whether the OS interface is up or down.
TCP/UDP/ICMP/Other	These lines report the type of packets received during matching. These numbers represent the type of protocol that was decided immediately after the IP header. In the case of IPv6, if an option is present after the header, "other" will be incremented.
Matched Packets	This line counts the total number of packets matched and captured by some policy.
Total Flows	This line reports the number of new flows started according to the flow policies.
Delayed Packets	If firmware has been compiled such that non-matching packets are not dropped (<*opt4*> in the firmware filename is set to “N”, see “Firmware Filenames” on page 46), this line reports the number of packets that were stored in a temporary buffer before their associated flow was captured. Otherwise, this counter shows the number of flows that are garbage-collected due to a timeout or a TCP session teardown.
Stateful Packets	This line reports the number of packets matched because of a stateful policy. The mathematical difference between this counter and the *Total Captured* counter is the number of packets captured by stateless policies.
Blocked Packets	This line reports the number of packets blocked because of rules.
Rx Packets/Bytes/Bits	These lines track data received by the OS. Any difference between the values in this line and those in the *Total Captured* line is due to buffering and/or packet loss; packet loss is due to high contention on the CPU.
Errors	This line reports the number of anomalous receive conditions the driver encounters. These conditions usually indicate packet loss due to high CPU usage.
Truncated Packets	This line reports the number of truncated packets received by the OS. Truncation is the result of malformed packets, the appliance settings, or a high load on the driver. A high load reduces the DPI buffer space; to compensate for this and avoid packet loss, the DPI truncates captured packets. Truncation always occurs after a pattern match.
Delayed Packets	This line reports the number of delayed packets received by the OS. Delayed packets are captured when a flow needs to be captured retroactively because of a stateful rule. When the dynamic rules are compiled with the stateful option, this line reports on the number of session teardowns. This is because packets are never captured retroactively when using the dynamic stateful option.

Reloading Firmware

During firmware reloading, all packets flow regardless of capture/forward policies, as the policies cannot be enforced during system initialization. This "open" state during configuration state transition ensures that there is no interruption of service when the DPI is updated.

If the operating system is halted or the device drivers are unloaded, the captured packets are stored in the DPI internal buffer (and mirrored in the P-Series P1), and the capture/forward policies are still enforced. Likewise, if the device drivers are rendered inactive by an OS crash, the appliance continues to operate independently.

Chapter 5 Command Line Interface

The command line interface (CLI) is an alternative to the GUI for managing the appliance. A script called *meta* is used to perform the same management functions as the GUI.

Invoke the meta script using the command syntax **meta** *command*; the OS environment variables are set such that this command can be executed from any path. The meta script commands are described in Table 6.

CLI Commands

Table 6 CLI Commands

Command	Description
meta compilerules	This command transforms the dynamic Snort rules contained in */usr/local/meta/0/rules.custom* into binary code suitable for the DPI processors.The binary code is stored in the file */usr/local/meta/meta_{0|1}.bin*. This process also updates the rule description databases */usr/local/meta/0/meta_{0|1}.custmapping*.
meta loadparams	This command uploads the runtime configuration parameters contained in the file */usr/local/meta/0/meta.conf*. The syntax of such parameter files is *(*<address>*)* <value> where <address> is the decimal address of the DPI control register, and <value> is the hexadecimal parameter to be loaded. Table 7 shows the parameters to which each address is mapped.
meta loadrules	This command uploads to the FPGA dynamic rules for both channels encoded in the files */usr/local/meta/0/meta_{0|1}.bin.* Capture/block policies previously stored are temporarily disabled during this operation and traffic is forwarded. The new rules take effect the loading process is complete.
meta off	This command disables the direct memory access (DMA). This might be a desirable state in traffic mirroring or pure filtering applications where the host is only used for control.
meta on	This command enables the DMA.
meta params	This command reads the currently stored parameters and runtime information in the DPI processor.
meta restart	This command performs the command **stop** followed by the command **start** as previously described. **Restart** always reloads the FPGA, as opposed to **start** which does not load the FPGA if firmware is already present.

Table 6 CLI Commands

Command	Description
meta start	This command: • Begins communication between the device driver and the appliance • Loads the firmware if firmware is not already present • Loads the capture/block configuration • Loads the runtime parameters and the dynamic rules • Loads the network interface device driver associated with the appliance, if not already loaded • Enables the network interface
meta stop	This command brings the appliance network interface down.
meta version	This command displays the driver version.

Table 7 shows the parameters to which each decimal address of the DPI control register is mapped.

Table 7 Loadparams Address Mapping

Address	Corresponding Parameter
Address 12	This address is mapped to the parameter *Flow timeout* (measured in seconds). This parameter controls how quickly the stateful packet analysis can garbage-collect previous states. Smaller values increase the number of concurrent flows that can be tracked. The default value is 16 seconds.
Address 16	This address is mapped to the parameter *Flow length* (measured in packets). This parameter controls the number of packets in a flow that are considered for capture and/or blocking. Typical values range from 6 to16.
Address 20	This address is mapped to the parameter *Truncation size* (measured in bytes). This parameter controls how many additional 16-bit words are captured after the last byte that triggers a matching event. Typical values are 0 or 1. A value of 0 means no truncation; 1 means maximum truncation.
Address 24	This address is mapped to the parameter *Burst size* (measured in bytes). This parameter sets the number of 32-bit words to transfer in one PCI master cycle. Larger bursts achieve higher throughput but may increase buffering latency and contention with other devices sharing the same bus. The default value is 1000.
Address 32	This address is mapped to the parameter *Inter-burst delay* (PCI cycles). This parameter can be used to reduce contention on the PCI bus caused by the line card. The default value is 5.

Editing Dynamic Rules with the CLI

Dynamic rules are stored in the file *rules.custom* in the */usr/local/meta/0* directory. To edit these rules:

Step	Task
1	Change directories to */usr/local/meta/0.*
2	Enter the command **vi rules.custom** to edit dynamic rules (see Appendix A, , on page 59 for information on *vi*).
3	Enter rules according to the format described in “Writing Rules” on page 49.
4	Save your changes and exit *vi*.
5	Enter **meta compilerules** to compile the new dynamic rules.
6	Enter **meta loadrules** upload the dynamic rules to the FPGA.

Selecting Firmware with the CLI

Firmware is composed of three files: one *.bit* file, and two *.mapping* files (one for each channel). Selecting firmware with the CLI requires manually linking the firmware’s *.bit* and *.mapping* files to corresponding target files in directory *0* (the GUI links these files for you).

To select firmware you must make the following symbolic links:

- Link the firmware *.bit* file to *meta.bit*
- Link the firmware *.mapping* file for Channel 0 to *meta_0.mapping*
- Link the firmware *.mapping* file for Channel 1 to *meta_1.mapping*

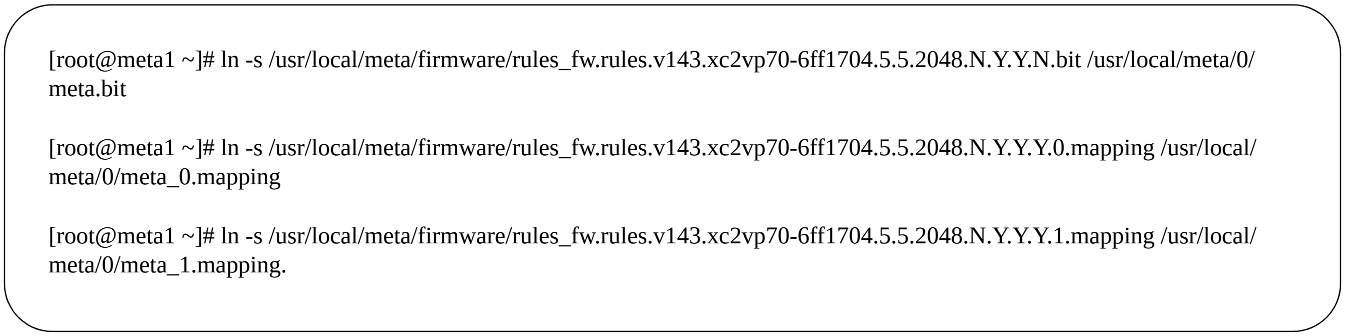
Files are symbolically linked with the **ln -s** command; the command syntax is as follows:

```
ln -s /usr/local/meta/0/firmware/file1 /usr/local/meta/0/file2
```

- *file1* represents the firmware *.bit* or *.mapping* files.
- *file2* represents *meta.bit, meta_0.mapping*, or *meta_1.mapping*.

Figure 18 illustrates this process.

Note: The firmware file name in Figure 18 is provided for example purposes. These file names must be substituted with the names of firmware files that are provided, or that you have compiled (see “Selecting Firmware with the GUI” on page 31).



```
[root@meta1 ~]# ln -s /usr/local/meta/firmware/rules_fw.rules.v143.xc2vp70-6ff1704.5.5.2048.N.Y.Y.N.bit /usr/local/meta/0/
meta.bit

[root@meta1 ~]# ln -s /usr/local/meta/firmware/rules_fw.rules.v143.xc2vp70-6ff1704.5.5.2048.N.Y.Y.Y.0.mapping /usr/local/
meta/0/meta_0.mapping

[root@meta1 ~]# ln -s /usr/local/meta/firmware/rules_fw.rules.v143.xc2vp70-6ff1704.5.5.2048.N.Y.Y.Y.1.mapping /usr/local/
meta/0/meta_1.mapping.
```

Figure 18 Linking Firmware Files

Linux Interface Counters

If the Linux command **ifconfig** is executed on the appliance interface, the displayed receive counters have modified meanings. These modified meanings are described in Table 8.

Table 8 Receive Counter Definitions

Counter Name	DPI Meaning
RX Packets	The total number of packets captured
RX Errors	The total number of packets that might have been dropped during the capturing process. The driver attempts to recover from these errors and might still provide the correct data
RX Dropped	The total number of packets blocked as a result of the capture/forward policies
RX Overruns	The total number of delayed packets captured as a result of the state transition
RX Frame	The total number of truncated packets captured
RX Bytes	The total number of captured bytes
Oerrs	The total number of captured bytes

Chapter 6 Compiling Rules

The *Meta Traffic Processor Compiler (MTP-Compiler)* produces user-defined firmware for the appliances. The user-defined input is a set of signature-based rules in Snort syntax, and compilation directives. The output of the compiler is a Xilinx bit file and ASCII mapping files that map specified signatures to internal configuration registers. The configuration registers are used to disable/enable rules or block packets.

Creating Rules Files

Store rules files in an *mtp-compiler* sub-directory — for example *mtp-compiler/rules*. Force 10 recommends not storing rules files elsewhere because this increases the length of the firmware file name.

Rules Capacity

The approximate maximum rules capacity of each appliance is given in Table 9. The maximum static rules capacity is 1000 and 650 for the P1 and P10, respectively; at maximum capacity, you are still allowed an additional 10 dynamic rules (with or without offsets). Alternatively, you have the option of using dynamic rules only; you are not obligated to use static rules.

Table 9 Maximum Rules Capacity

Model	Static and Dynamic Rules	Dynamic Rules Only
P-Series P1	1000 Static + 10 Dynamic	100
P-Series P10	650 Static + 10 Dynamic	70

The space required for a static rule depends upon its complexity, but in general, static rules take up less space than dynamic rules, which are inherently complex.

- 1 dynamic rule with offsets take the space of approximately 10 static rules.
- 1 dynamic rule without offsets take the space of approximately 5 static rules.

Less of one type of rule allows more of the other. If you use less static rules, for example, then you are able to fit more dynamic rules by a proportional factor. The approximate capacity of dynamic rules with offsets (D_{offset}) in relation to the number of static rules can be determined by Equation 1. Likewise, Equation 2 determines the approximate capacity for dynamic rules without offsets (D).

$$D_{offset} = \frac{(S_{max} - S)}{10} \qquad Equation\ 1$$

$$D = \frac{(S_{max} - S)}{5} \qquad Equation\ 2$$

S_{max} is the maximum number of static rules given by Table 9, and S is the number of static rules in your rules file (a count of the number of rules in the rules file is given during the configuration phase of the compilation process).

Compiling Rules

Note: The MTP-Compiler is managed with GNU make.

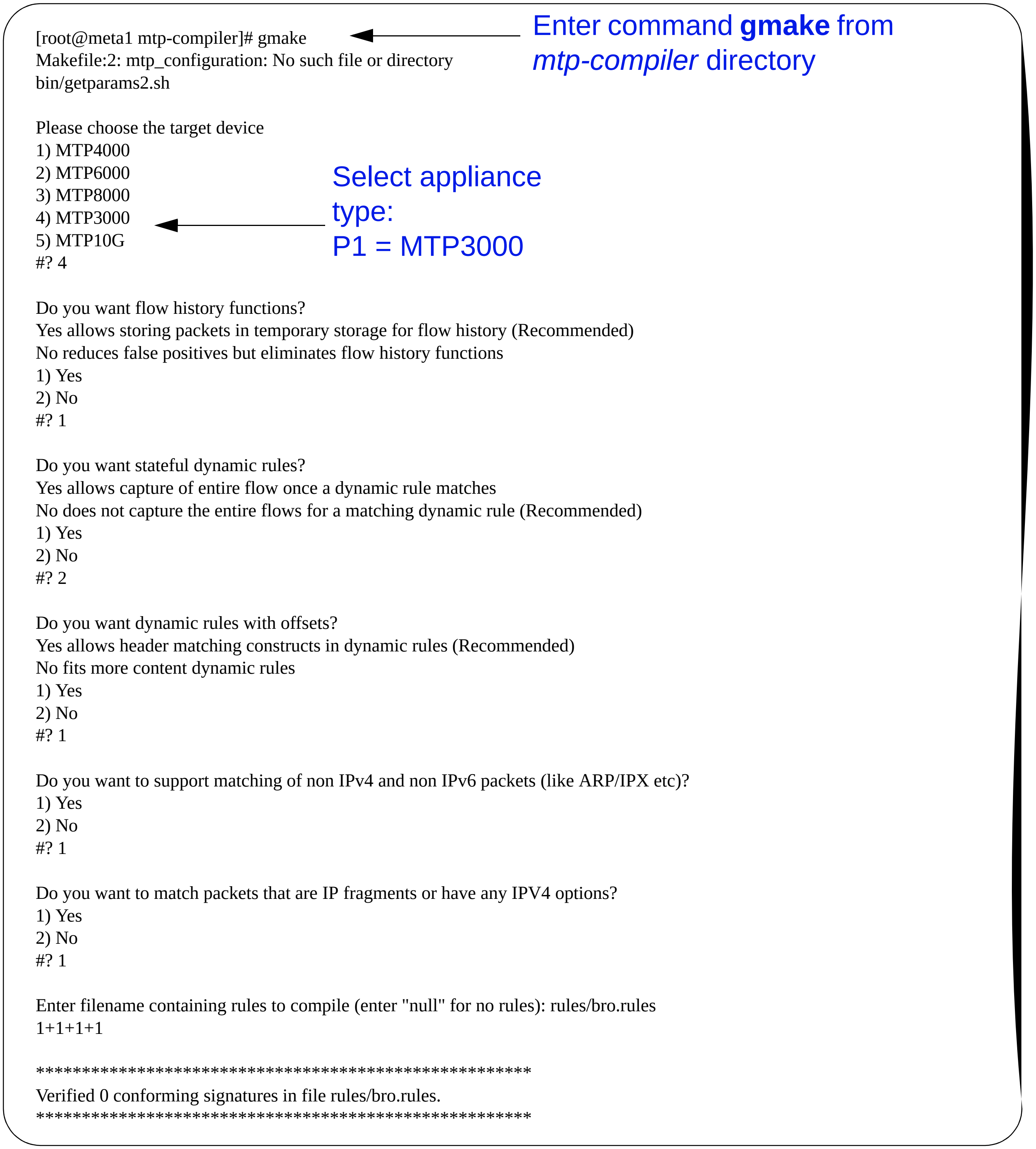
1. Change directory to *mtp-compiler*. Enter the command **gmake**. This command invokes the configuration script, the MTP compiler, and the Xilinx compiler, in succession. Entering **time gmake** invokes the same processes, but this command measures the compilation time as well.
2. The script prompts you for a number of compilation options. Refer to Table 10 for a description of each option, and enter a response for each.

Table 10 Compiler Configuration Options

	Compilation Option	Description
1	Target Device	Choose the type of firmware to generate based on the model of the line card. • The P1 requires type **MTP3000** • The P10 requires type **MTP10G** (see Figure 19 on page 42)
2	Store Flow History	Answering **Yes** allows storing packets in temporary storage for flow history Answering **No** reduces false positives but eliminates flow history functions
3	Stateful Dynamic Rules	Answering **Yes** to this option allows the capture of an entire flow matching any of the dynamic rules. If a dynamic rule triggers, all the packets mapped to that flow will be captured until a timeout occurs, or the flow terminates with a FIN or RST packet (see Figure 19 on page 42).
4	Dynamic Rules with Offsets	Answering **Yes** to this option allows header matching constructs in dynamic rules. Answering **No** disables offsets; in this case, dynamic rules will only match on the content of IPv4 and IPv6 packets (see Figure 19 on page 42).
5	Match non-IP Traffic	Answering **Yes** to this option matches packets which are neither IPv4 nor IPv6. This option should be set to **Yes** if only IP traffic is allowed.
6	Match Fragmented IPv4 Packets or IPv4 Packets w/ Options	Answering **Yes** to this option: • Adds a rule to match fragmented IPv4 packets • Adds a rule to match IPv4 packets with any option in the header (see Figure 19 on page 42).

Table 10 Compiler Configuration Options

	Compilation Option	Description
7	Rules File	Specify the rules file that contains the Snort rules that will be compiled into firmware. • Include the relative path of the file in your entry. • Your entry is used to create the firmware names. • Enter **null** to create firmware with no static rules; compiling firmware with no static rules, maximizes dynamic rule capacity (see Figure 20 on page 43). **Note:** The script performs a syntax check on the input file. If there are errors, you are prompted to enter the file name again. The entry must be made at the prompt; if the *Enter* key is pressed erroneously such that the entry cannot made at the prompt, enter **Ctrl-C** to halt the configuration process, and then enter **gmake** to begin again.
8	Dynamic Rules	Enter the number of dynamic rules to synthesize. • If you enter one of the sample Snort rules files, choose the minimum number of dynamic rules; otherwise, the placing may fail. • If you are using fewer static rules, you can increase the number of dynamic rules up to a total of approximately 60 for the P-Series P10 and 70 for the P1 (see Figure 20 on page 43).
9	Meta Rules	The MTP-Compiler prepends a set of fixed rules called *rule.meta* — located in the *mtp-compiler/rules* directory; include or exclude this file. The rules in this file are displayed, and they report on flow information and handle possible TCP segmentation evasion attempts. They provide compatibility with Snort, and including them allows you to run Snort on the DPI interface. It is best to include this file if Snort is being used as the front end. If not using Snort as the front end, these rules should not be included or they should be changed to accommodate other packet analysis requirements (see Figure 21 on page 44).
10	Maximum String	Specify the maximum number of bytes a single static rule can use for content matching. A low value truncates the match string and increases the number of rules that can fit into the FPGA, but this is at the expense of increased false positives. A value lower than 1024 is not recommended unless you can cope with the increased number of false positives through Snort or some other means (see Figure 21 on page 44).
11	Confirmation	Enter **Yes** to save the configuration and compile the Snort rules into firmware (see Figure 22 on page 45).

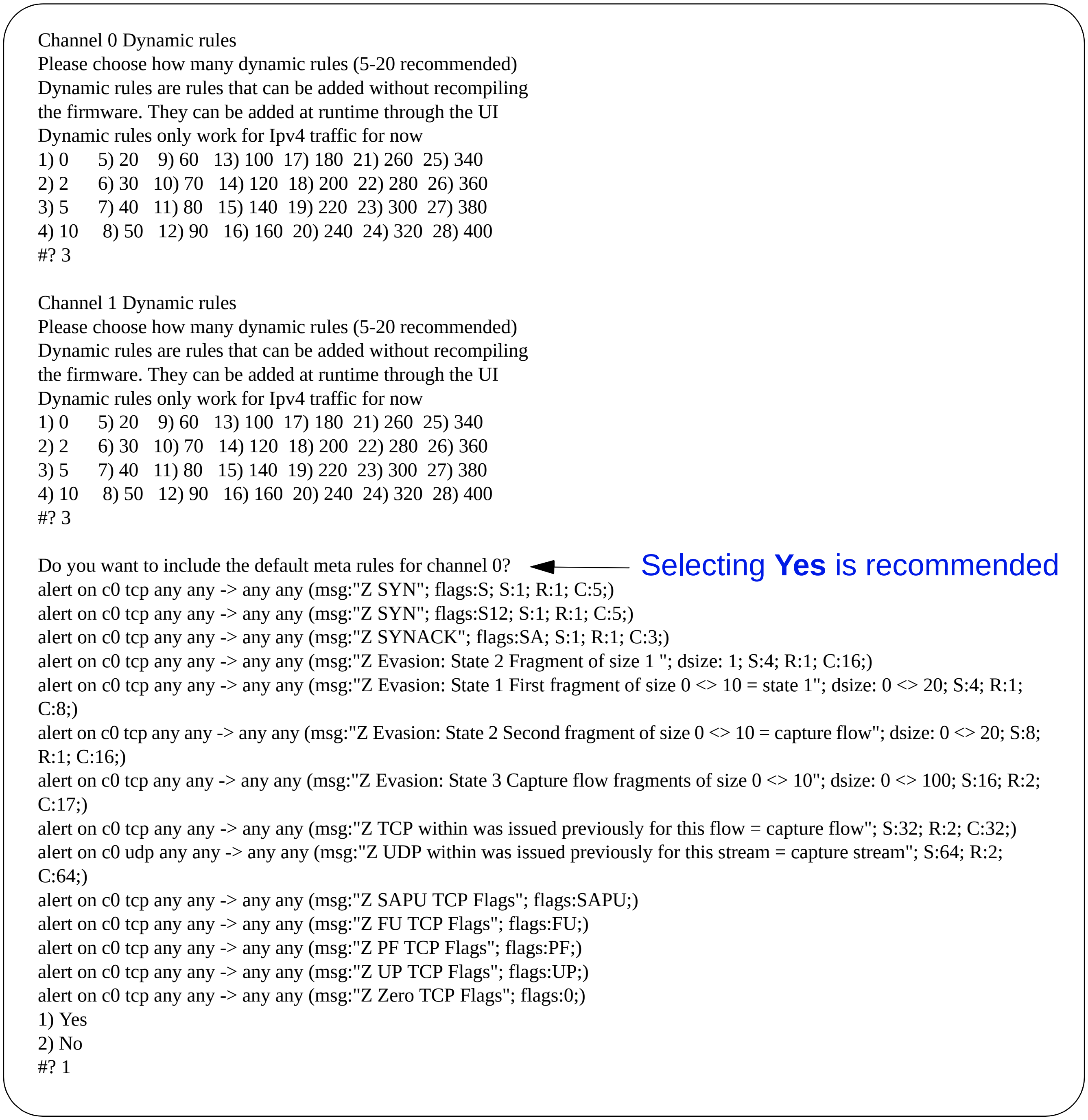


```
[root@meta1 mtp-compiler]# gmake
Makefile:2: mtp_configuration: No such file or directory
bin/getparams2.sh

Please choose the target device
1) MTP4000
2) MTP6000
3) MTP8000
4) MTP3000
5) MTP10G
#? 4

Do you want flow history functions?
Yes allows storing packets in temporary storage for flow history (Recommended)
No reduces false positives but eliminates flow history functions
1) Yes
2) No
#? 1

Do you want stateful dynamic rules?
Yes allows capture of entire flow once a dynamic rule matches
No does not capture the entire flows for a matching dynamic rule (Recommended)
1) Yes
2) No
#? 2

Do you want dynamic rules with offsets?
Yes allows header matching constructs in dynamic rules (Recommended)
No fits more content dynamic rules
1) Yes
2) No
#? 1

Do you want to support matching of non IPv4 and non IPv6 packets (like ARP/IPX etc)?
1) Yes
2) No
#? 1

Do you want to match packets that are IP fragments or have any IPV4 options?
1) Yes
2) No
#? 1

Enter filename containing rules to compile (enter "null" for no rules): rules/bro.rules
1+1+1+1

*******************************************************
Verified 0 conforming signatures in file rules/bro.rules.
*******************************************************
```

Enter command **gmake** from *mtp-compiler* directory

Select appliance type:
P1 = MTP3000

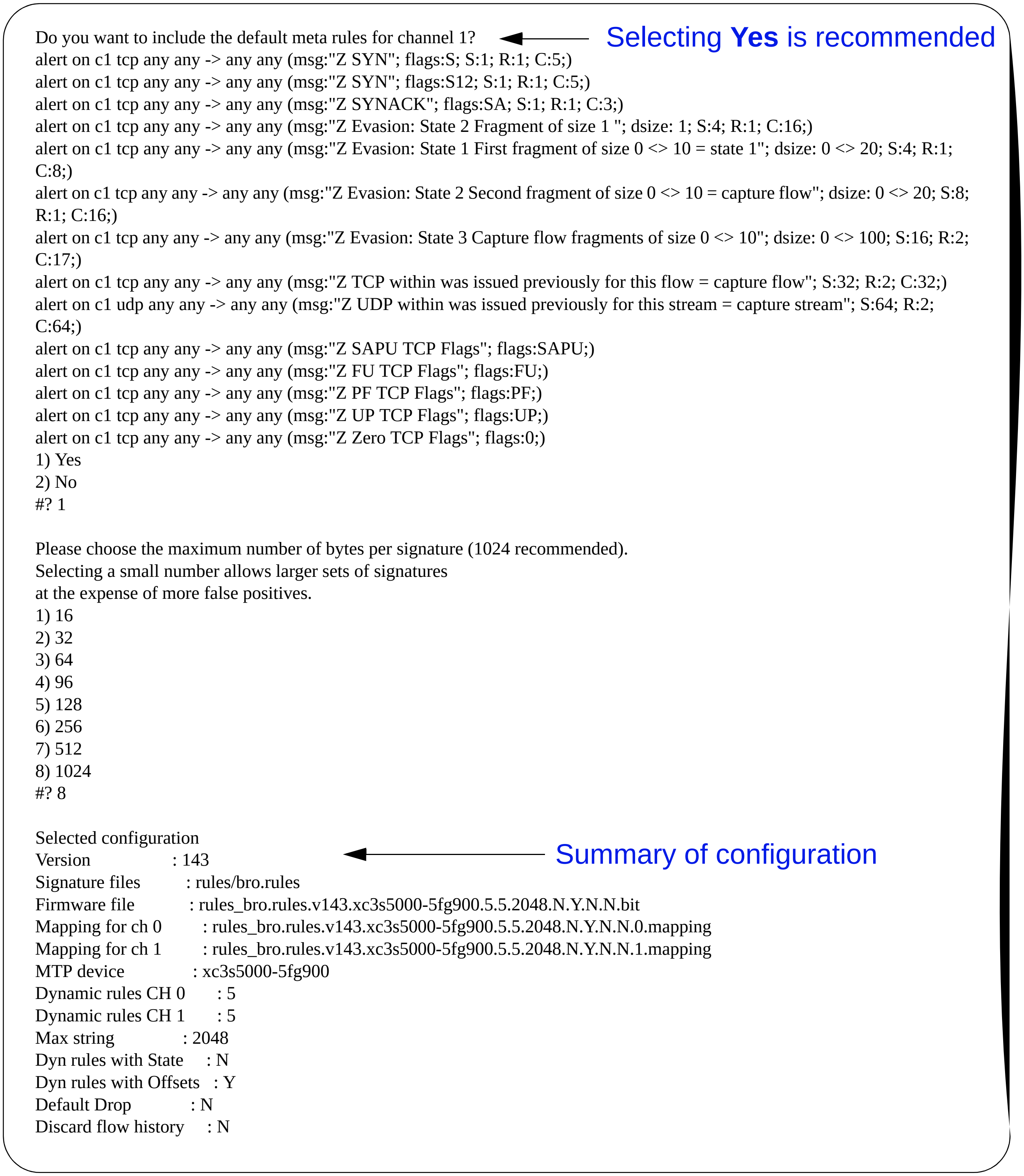
Figure 19 MTP-Compiler Option 1-7



```
Channel 0 Dynamic rules
Please choose how many dynamic rules (5-20 recommended)
Dynamic rules are rules that can be added without recompiling
the firmware. They can be added at runtime through the UI
Dynamic rules only work for Ipv4 traffic for now
1) 0      5) 20    9) 60   13) 100  17) 180  21) 260  25) 340
2) 2      6) 30   10) 70   14) 120  18) 200  22) 280  26) 360
3) 5      7) 40   11) 80   15) 140  19) 220  23) 300  27) 380
4) 10     8) 50   12) 90   16) 160  20) 240  24) 320  28) 400
#? 3

Channel 1 Dynamic rules
Please choose how many dynamic rules (5-20 recommended)
Dynamic rules are rules that can be added without recompiling
the firmware. They can be added at runtime through the UI
Dynamic rules only work for Ipv4 traffic for now
1) 0      5) 20    9) 60   13) 100  17) 180  21) 260  25) 340
2) 2      6) 30   10) 70   14) 120  18) 200  22) 280  26) 360
3) 5      7) 40   11) 80   15) 140  19) 220  23) 300  27) 380
4) 10     8) 50   12) 90   16) 160  20) 240  24) 320  28) 400
#? 3

Do you want to include the default meta rules for channel 0?
alert on c0 tcp any any -> any any (msg:"Z SYN"; flags:S; S:1; R:1; C:5;)
alert on c0 tcp any any -> any any (msg:"Z SYN"; flags:S12; S:1; R:1; C:5;)
alert on c0 tcp any any -> any any (msg:"Z SYNACK"; flags:SA; S:1; R:1; C:3;)
alert on c0 tcp any any -> any any (msg:"Z Evasion: State 2 Fragment of size 1 "; dsize: 1; S:4; R:1; C:16;)
alert on c0 tcp any any -> any any (msg:"Z Evasion: State 1 First fragment of size 0 <> 10 = state 1"; dsize: 0 <> 20; S:4; R:1;
C:8;)
alert on c0 tcp any any -> any any (msg:"Z Evasion: State 2 Second fragment of size 0 <> 10 = capture flow"; dsize: 0 <> 20; S:8;
R:1; C:16;)
alert on c0 tcp any any -> any any (msg:"Z Evasion: State 3 Capture flow fragments of size 0 <> 10"; dsize: 0 <> 100; S:16; R:2;
C:17;)
alert on c0 tcp any any -> any any (msg:"Z TCP within was issued previously for this flow = capture flow"; S:32; R:2; C:32;)
alert on c0 udp any any -> any any (msg:"Z UDP within was issued previously for this stream = capture stream"; S:64; R:2;
C:64;)
alert on c0 tcp any any -> any any (msg:"Z SAPU TCP Flags"; flags:SAPU;)
alert on c0 tcp any any -> any any (msg:"Z FU TCP Flags"; flags:FU;)
alert on c0 tcp any any -> any any (msg:"Z PF TCP Flags"; flags:PF;)
alert on c0 tcp any any -> any any (msg:"Z UP TCP Flags"; flags:UP;)
alert on c0 tcp any any -> any any (msg:"Z Zero TCP Flags"; flags:0;)
1) Yes
2) No
#? 1
```

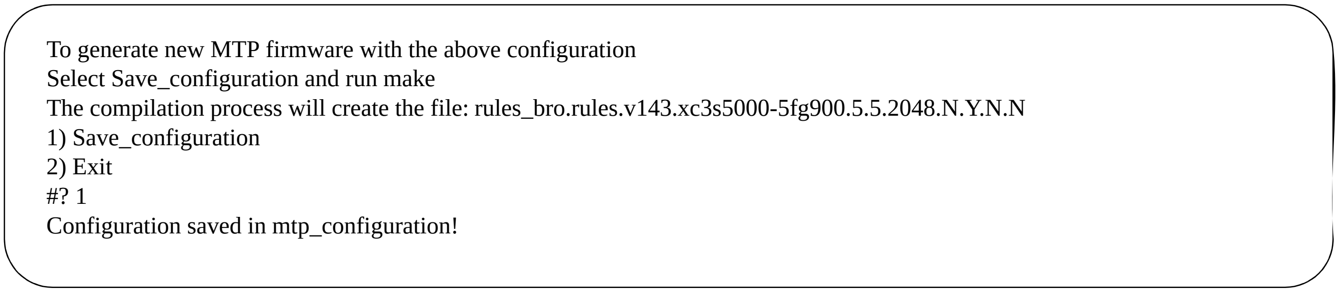
Selecting **Yes** is recommended

Figure 20 MTP-Compiler Option 8-9



```
Do you want to include the default meta rules for channel 1?
alert on c1 tcp any any -> any any (msg:"Z SYN"; flags:S; S:1; R:1; C:5;)
alert on c1 tcp any any -> any any (msg:"Z SYN"; flags:S12; S:1; R:1; C:5;)
alert on c1 tcp any any -> any any (msg:"Z SYNACK"; flags:SA; S:1; R:1; C:3;)
alert on c1 tcp any any -> any any (msg:"Z Evasion: State 2 Fragment of size 1 "; dsize: 1; S:4; R:1; C:16;)
alert on c1 tcp any any -> any any (msg:"Z Evasion: State 1 First fragment of size 0 <> 10 = state 1"; dsize: 0 <> 20; S:4; R:1;
C:8;)
alert on c1 tcp any any -> any any (msg:"Z Evasion: State 2 Second fragment of size 0 <> 10 = capture flow"; dsize: 0 <> 20; S:8;
R:1; C:16;)
alert on c1 tcp any any -> any any (msg:"Z Evasion: State 3 Capture flow fragments of size 0 <> 10"; dsize: 0 <> 100; S:16; R:2;
C:17;)
alert on c1 tcp any any -> any any (msg:"Z TCP within was issued previously for this flow = capture flow"; S:32; R:2; C:32;)
alert on c1 udp any any -> any any (msg:"Z UDP within was issued previously for this stream = capture stream"; S:64; R:2;
C:64;)
alert on c1 tcp any any -> any any (msg:"Z SAPU TCP Flags"; flags:SAPU;)
alert on c1 tcp any any -> any any (msg:"Z FU TCP Flags"; flags:FU;)
alert on c1 tcp any any -> any any (msg:"Z PF TCP Flags"; flags:PF;)
alert on c1 tcp any any -> any any (msg:"Z UP TCP Flags"; flags:UP;)
alert on c1 tcp any any -> any any (msg:"Z Zero TCP Flags"; flags:0;)
1) Yes
2) No
#? 1

Please choose the maximum number of bytes per signature (1024 recommended).
Selecting a small number allows larger sets of signatures
at the expense of more false positives.
1) 16
2) 32
3) 64
4) 96
5) 128
6) 256
7) 512
8) 1024
#? 8

Selected configuration
Version               : 143
Signature files       : rules/bro.rules
Firmware file         : rules_bro.rules.v143.xc3s5000-5fg900.5.5.2048.N.Y.N.N.bit
Mapping for ch 0      : rules_bro.rules.v143.xc3s5000-5fg900.5.5.2048.N.Y.N.N.0.mapping
Mapping for ch 1      : rules_bro.rules.v143.xc3s5000-5fg900.5.5.2048.N.Y.N.N.1.mapping
MTP device            : xc3s5000-5fg900
Dynamic rules CH 0    : 5
Dynamic rules CH 1    : 5
Max string            : 2048
Dyn rules with State  : N
Dyn rules with Offsets : Y
Default Drop          : N
Discard flow history  : N
```

Figure 21 MTP-Compiler Option 10-11



```
To generate new MTP firmware with the above configuration
Select Save_configuration and run make
The compilation process will create the file: rules_bro.rules.v143.xc3s5000-5fg900.5.5.2048.N.Y.N.N
1) Save_configuration
2) Exit
#? 1
Configuration saved in mtp_configuration!
```

Figure 22 MTP-Compiler Option 12

Starting and Stopping the MTP-Compiler

Enter the keyboard command **Ctrl-C** or a *SIGINT* signal to interrupt the compilation or configuration process. Enter **gmake** to restart the process from where it was interrupted. The compilation process restarts at the point where it was halted; the configuration process restarts from the beginning.

During compilation, enter **Ctrl-C** followed by **gmake clean** to regenerate firmware with different options. This erases the current configuration and resets the compilation process. Previously generated firmware files will not be erased.

Configuration and Generated Files

Table 11 describes the files that are used or generated by the MTP-Compiler.

Table 11 Configuration and Generated Files

File	Description	Location
<firmware filename>.bit	This file is generated after compiling static rules. When selecting firmware, this file must be symbolically linked to the file *meta.bit*.	/usr/local/meta/firmware
<firmware filename>_{0|1}.mapping	These files are generated after compiling static rules. When selecting firmware, this file must be symbolically linked to *meta_{0|1}.mapping*.	/usr/local/meta/firmware
meta.bit	This is the file to which *<firmware filename>.bit* must be symbolically linked when selecting firmware.	/usr/local/meta/0
meta.conf	This file contains the runtime configuration parameters. These parameters are displayed at the top of the Runtime Statistics screen.	/usr/local/meta/0
meta_{0|1}.bin	These files contain compiled dynamic rules for Channel 0 and Channel 1.	/usr/local/meta/0

Table 11 Configuration and Generated Files

File	Description	Location
meta_{0|1}.custmapping	These files contain the capture/forward policies for each rule on Channel 0 and Channel 1.	/usr/local/meta/0
meta_{0|1}.mapping	This is the file to which *<firmware filename>_0.mapping* must be symbolically linked when selecting firmware.	/usr/local/meta/0
rules.custom	This file contains dynamic rules written in Snort syntax.	/usr/local/meta/0

Firmware Filenames

The MTP-Compiler creates new firmware — in the */usr/local/meta/firmware* directory — consisting of one *.bit* file and two *.mapping* files; one mapping file is for Channel 0 and the other is for Channel 1.

Firmware filenames follow a naming convention designed to identify three properties:

- The appliance that can use it
- The number of dynamic rules
- The maximum allowed number of half-bytes per rule

Firmware files have the format:

```
<name>.<type>.<dynamic0|1>.<maxstring>.<opt1>.<opt2>.<opt3>.<opt4>.<opt5>.{bit|mapping}
```

Table 12 describes each of the elements in this format.

Table 12 Firmware Filename Description

Element	Description
<name>	This field is a mnemonic name identifying the original rules file you supplied during the compilation of the firmware.
<type>	This field identifies the card type. The P1 appliance is represented by *xc3s5000-4fg900-2,* and the P10 is represented by *xc2vp70-6ff1704.*
<dynamic0|1>	This field is the estimated number of dynamic rules that you can enter at runtime for the two channels.
<maxstring>	This field is the maximum number of half-bytes the compiler allocates for each rule. A typical value is 2048 to indicate that the compiler truncates match string to 1024 bytes. Typically a value is 2048, which does not result in any truncation. Lower values are possible and result in a larger number of rules, but this increases the probability of false positives for rules with truncated match strings.

Table 12 Firmware Filename Description

Element	Description
<opt1>	"Y" or "N" is listed in this field depending upon whether or not dynamic rules were compiled with flow capture support. The default is "N". **Note:** This field is not valid for the P10 appliance; "N" is always listed.
<opt2>	"Y" or "N" is listed in this field depending upon whether or not dynamic rules were compiled with offset support. The default is "Y". **Note:** This field is not valid for the P10 appliance; "N" is always listed.
<opt3>	"Y" or "N" is listed in this field depending upon whether or not the firmware has been compiled such that non-IP/IPv6 packets are allowed through. See "Compiler Configuration Options" on page 40.
<opt4>	"Y" or "N" is listed in this field depending upon whether or not the firmware has been compiled such that non-matching packets are dropped. See "Compiler Configuration Options" on page 40.
<opt5>	"Y" or "N" is listed in this field depending upon whether or not state memory is used to hold a hash key. See "Compiler Configuration Options" on page 40. • For Snort compatibility this option should be set to "N." • If this options is set to "N", the driver can access temporary memory to retrieve flow history information stored with the R=1 flag. • If this options is set to "Y", and additional 32-bit key is stored into state memory to reduce false positives in stateful IDS rules.
{bit|mapping}	The compiling process generates three files which together make firmware. Two files have the extension *.mapping*, and one has the extension *.bit*

Compiler Errors

- If too many dynamic rules are specified in Option 9 of the compiler configuration phase, the compilation process will fail, and you will receive a "Error-PhysDesignRules" error message. In this case, enter **gmake clean** to erase the current configuration and begin again.
- If too many rules stored in the rules file specified in Option 6 of the compiler configuration phase, the compilation process will fail. In this case, enter **gmake clean** to erase the current configuration and begin again.

Chapter 7 Writing Rules

Rule Syntax

Rules have the following syntax:

```
<capture/forward policy> on <channel> <Snort rule>
```

- <capture/forward policy> can have four values: *alert*, *permit*, *divert*, or *deny*. These settings are described in Table 4 on page 28.
- <channel> can be “c0” for Channel 0 or “c1” for Channel 1.
- <Snort rule> is a rule written in Snort syntax.

Table 13 shows an example rule.

Table 13 Rule Syntax

alert on c1 any any -> any any (msg:"Z Default rule fragmented ip";)

Snort Supported Constructs

Table 14 lists Snort constructs that the P-Series supports for both dynamic and static rules.

Note: Rules using non-supported constructs can be compiled, but they may produce ambiguous results. Constructs denoted with a “*” are not supported by the P10.

Table 14 Supported Snort Constructs for Static and Dynamic Rules

Construct	Static	Dynamic	Construct	Static	Dynamic
ack	Yes	Yes	seq	Yes	Yes
case	Yes	No	source/ destination address	Yes	Only /8/16/24/32 masks
content/ uricontent	Yes, no negative.	No	source/ destination port	Yes	Yes, no ranges
dsize	Yes	No	tos	Yes	Yes
flags	Yes	Yes, no wild card	ttl	Yes	Yes
flow	Yes	No	ttl6*	Yes	No
icmp id	Yes	Yes	byte_jump	No	No
icmp seq	Yes	Yes	byte_test	No	No
icode	Yes	Yes	content-list	No	No
icode6*	Yes	No	distance	No	No
id	Yes	Yes	frag bits	No	No
ip proto	Yes	Yes	fragoffset	No	No
ipv6*	Yes	No	ip options	No	No
ipv6 src/ dst address*	Only masks which are multiples of 8	No	offset	No	No
ipv6ah*	Yes	No	resp	No	No
ipv6dest*	Yes	No	rpc	No	No
ipv6hbyh*	Yes	No	same ip	No	No
itype	Yes	Yes	session	No	No
itype6*	Yes	No	tag	No	No
payload	Yes	No	within	No	No
proto	ICMP, UDP, TCP, ICMPv6, IP, IPv6	ICMP, UDP, TCP, IP			

IPv6 Snort Constructs

The P-Series requires special, Force 10 Snort constructs to handle IPv6. These are also included in Table 14, and examples of rules using IPv6 constructs are given in Table 15.

Table 15 Example Rules Using the IPv6 Construct

alert on c0 tcp any any -> any any (msg:"Hop by hop with 000102 and payload livio case sensitive"; ipv6hbyh; content:"|000102|"; payload; content:"livio"; classtype:attempted-recon; sid:1213; rev:4;)
alert on c0 tcp any any -> any any (msg:"Hop by hop with 000102 and payload livio case insensitive"; ipv6hbyh; content:"|000102|"; payload; content:"livio"; nocase; classtype:attempted-recon; sid:1213; rev:4;)

Note: The Snort packet capture application does not support Ipv6; therefore an alternate packet capture application must be used if using these constructs.

ipv6hbyh, *ipv6dest*, *ipv6*, *route*, *ipv6ah*, and *payload* are context operators. They must be followed by a content operator that can be matched within the given context.

Table 16 describes the content to which the context operators refer. If no context is specified (none of the keywords is present in the rule) the context defaults to *payload*.

Table 16 Context and Content Operators

Context Operator	Corresponding Content Operator
ipv6hbyh	Matches the following content in the IPv6 Hop by Hop option.
ipv6dest	Matches the following content in the IPv6 Destination option.
ipv6route	Matches the following content in the IPv6 Routing option.
ipv6ah	Matches the following content in the IPv6 Authentication option.
ipv6ah	Matches the following content in the IPv6 Security option.
payload	Matches the following content in the user payload.

Multiple content strings following a context operator would match the specified strings starting at any location and separated by any number of characters. The case modifier can apply to the IPv6 options as well.

An example of the IP protocol for both IPv4 and IPv6 is given in Table 17. These rules will match all IPv4 and IPv6 packets.

Table 17 Example Rules Using the IP Protocol

alert on c0 ip any any -> any any (msg:"any ip packet");
alert on c0 IPv6 any any -> any any (msg:"any ipv6 packet");

If port numbers are included, it is necessary to specify UDP or TCP; specifying IP will result in erroneous behavior (see Table 18).

Table 18 Example Rules Using Port Numbers

alert on c0 tcp any any -> any 80 (msg:"any tcp packet with dst port 80");
alert on c0 udp any any -> any 80 (msg:"any udp packet with dst port 80");

Writing Stateful Rules

Stateful matching improves the accuracy of detection because it adds ordering when specifying behaviors across multiple matching events. State transitions in the P-Series follow a non-cyclic pattern; no state transitions may erase any of the previous states. New state transitions are simply recorded via a non-destructive, additive operation.

As new states are produced, they are bitwise “*OR*-ed” with the current states contained in the per-flow register C_f. This method is different from stateful matching in software systems, where old state is removed after a set amount of time. It allows a deterministic wire-speed state management algorithm while guaranteeing that no match events are ever lost due to resource constraints.

Figure 23 shows the state matching algorithm. Note that the only time some state is erased is in the case of a timeout.

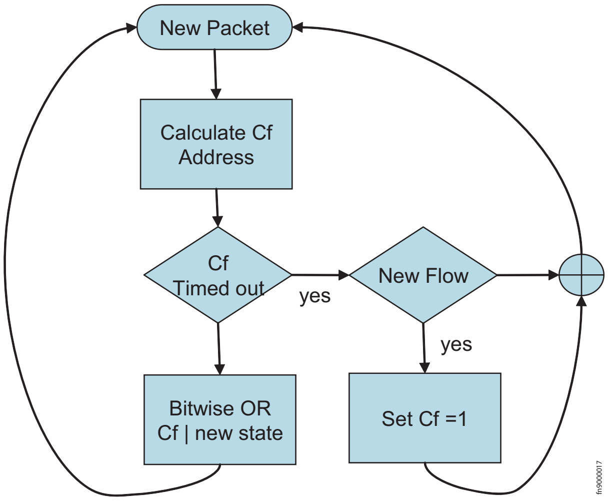


Figure 23 State Management Algorithm

Stateful Matching

Each signature *i* contains a pattern matching expression m_i that is compared to the incoming data stream in real time (time *t*). In addition, each signature may contain - at your discretion - three values, *s*, *c*, and *r*, which respectively specify:

- The pre-match state condition necessary for the signature to match (in addition to m_i)
- The post-match state condition applied after the signature has matched
- A directive indicating what to do with the matched packet

The *s* and *c* values are used to manage a per-flow register C_f, where the subscript *f* is the flow, or *sub-stream,* and the *r* value is used to direct the packet storage.

Pre-match Condition - the S Value

The value in register C_f is presented to all the signatures simultaneously during matching.

C_f must have all the bits specified by s_i (in addition to matching m_i) in order for the signature *i* to trigger. In other words, if the result of the logical "AND" of register C_f with s_i is non-zero and equal to s_i, the signature is allowed to trigger. Otherwise the signature is not triggered. Therefore value s_i is referred to as the pre-match bit pattern.

Post-match Condition - the C Value

The c_i value is the post-match bit pattern defined by the signature *i*. If m_i matches in the data stream, and the pre-match condition is met, c_i is logically "*OR*-ed" with the existing value in register C_f, and the result is written back to C_f.

In general for each signature *i* at time *t:*

$$If\ \left\{m_i \wedge (s_i^t\ \&\ C_f^{t-1})\right\},\ then\ cp_i^t = c_i,\ else\left\{cp_i^t = 0\right\} \qquad Equation\ 3$$

$$C_f^{t+1} = \sum cp_i^t \qquad Equation\ 4$$

where /\ is a logical "AND" operator and *&* is a bitwise "AND" operator.

Equation 3 states that if there is a match $m_{i,}$ and the pre-match condition holds, the post-match condition cp_i is enabled.

Equation 4 states that at each cycle, the register C_f is updated by the bitwise "OR" of all the cp_i values of all the signatures.

A special pattern - *s* is equal to 1 - will erase and rewrite the C_f register if the last access to the C_f register is older than a timeout value, to indicate the end of useful state information. Sixty seconds is the usual timeout used to reclaim stateful data since it is the typical timeout used by TCP endpoints.

Packet Handling - the R Value

The constant r_i is a flag that tells the hardware what to do with a packet that has been matched to signature *i*. The memory used to store the matched packets is divided into *Temporary Memory* and *Match Memory*. If a packet is stored in Match Memory, action is requested from the host to process the matched packet. If a packet is stored in Temporary Memory, no action is requested from the host, as this represents only a partial match.

When a packet is stored in either Temporary Memory or Match Memory, a pointer to the previously stored packet in the same flow (contained in a portion of the flow register C_f) is also stored. Thus a packet stored in Match Memory may reference another packet stored in Temporary Memory, which in turn may reference more packets, thus forming a linked list of partial matches, starting with a packet stored in Match Memory.

The values for r_i have the following meanings:

0: do not store the packet; only update the C_f register

1: store the packet in Temporary Memory

2: store the packet in Match Memory and notify host software

Note: If the Hash key option is selected, the *R=2* flag no longer causes the packet to be stored in Temporary Memory.

Stateful Rule Examples

Table 19 Stateful Matching Signatures

alert on c0 tcp any any -> any any (msg:"SYN"; flags:S; S:1; R:0; C:3;)
alert on c0 tcp any any -> any any (msg:"ack"; flags:A+; S:2; R:1; C:4;)
alert on c0 tcp any any -> any any (msg:"ack"; flags:A+; S:4; R:2; C:4;)
alert on c0 tcp any any -> any any (msg:"frag"; dsize: 0 <> 100; S:1; R:1; C:9;)
alert on c0 tcp any any -> any any (msg:"frag"; dsize: 0 <> 100; S:8; R:1; C:16;)
alert on c0 tcp any any -> any any (msg:"frag"; dsize: 0 <> 100; S:16; R:2; C:16;)

In Table 19:

- Signature 1 matches any TCP SYN packet, erasing any expired C_f register; if this signatures triggers - meaning a SYN is present — it will set bits 0 and 1 (value 3) in the C_f register. The SYN packets will be discarded (R=0).
- Signature 2 will trigger if Signature 1 has triggered (the C_f register having bit 1 set) and a TCP packet contains an ACK bit. The result for this match is that bit 2 (value 4) is set in the $\mathrm{C_f}$ register. The packet will be stored in Temporary Memory (R=1).
- Signature 3 will trigger if Signature 2 has triggered (the C_f register having bit 2 (value 4) set) and another later TCP packet contains an ACK bit. The result for this match does not modify the existing content of the C_f register. The packet will be stored in Match Memory, referencing the packet of Signature 2. The DPI driver will then present to the host the packet matched by 2, followed by the packet matched by 3, through the DPI network interface.

You can inspect Signatures 4, 5, and 6, and verify that they trigger a match and place a packet in Match Memory — thus alerting the host — if three consecutive packets are seen with size between 0 and 100. The third packet references the previous two stored in Temporary Memory. Thus, once the third packet is received, the three segments are presented to the host through the DPI network interface. Notice that the bit pattern used in the two rules avoids collision with the previous rule if the flow hashing also happens to collide.

The *rules.meta* File

The *rules.meta* file — located in the *mtp-compiler/rules* directory — specifies a number of stateful rules to be used with standard Snort rules (which use the *Flow* construct). In addition, these rules implement a stateful mechanism to circumvent some common forms of TCP IDS evasion.

Support for Snort's *flow* Construct

The two stateful rules in Table 20 initiate a new flow if a *SYN* or a *SYN-ACK* are seen. A Snort *flow-established* construct is translated to S:4 and S:2 for client-to-server and server-to-client flows, respectively. These constructs are automatically inserted by the MTP-Compiler when a flow-established construct is encountered during compilation. You can also insert the constructs directly into your rules.

Table 20 Flow Established Rules

alert on c0 tcp any any -> any any (msg:"Z SYN"; flags:S; S:1; R:1; C:5;)
alert on c0 tcp any any -> any any (msg:"Z SYNACK"; flags:SA; S:1; R:1; C:3;)

Handling Segmentation Evasion

Tools like *fragroute* or *Nessus* are used to fragment the packet payload in several TCP segments in order to evade packet-based signature systems. The stateful rules in Table 21 detect the arrival of packets exhibiting an anomalous use of TCP segmentation.

The start of the state machine is prompted by a *SYN*; state 1 is reached if a packet of length greater than 0 but less than 20 is detected; state 2 is reached if a packet of length 1 is received right after a SYN or a second packet of length greater than 0 but less than 20 is detected; the final state is reached if a packet of a length between 0 and 100 is seen. This state diagram was derived from observing common fragmentation evasion patterns; it seems to catch most of them. More complex state diagrams can also be devised at your discretion.

Table 21 TCP Packets with Anomalous Segmentation

alert on c0 tcp any any -> any any (msg:"Z Evasion: State 2 Fragment of size 1 "; dsize: 1; S:4; R:1; C:16;)
alert on c0 tcp any any -> any any (msg:"Z Evasion: State 1 First fragment of size 0 <> 20 = state 1"; dsize: 0 <> 20; S:4; R:1; C:8;)
alert on c0 tcp any any -> any any (msg:"Z Evasion: State 2 Second fragment of size 0 <> 20 = capture flow"; dsize: 0 <> 20; S:8; R:1; C:16;)
alert on c0 tcp any any -> any any (msg:"Z Evasion: State 3 Capture flow fragments of size 0 <> 100"; dsize: 0 <> 100; S:16; R:2; C:16;)

Support for Snort's *within* Construct

Many buffer-overflow detection rules use a *within* construct that verifies that an end-of-line character is received within a certain number of bytes from the start of the session.

If the *within* statement is for a large number of bytes, the check needs to be performed across TCP segments. In this case, several packets must be captured to find the end-of-line character (or whatever the character might be). For this reason, *within* statements capture the entire flow.

The *within* statements are translated by the MTP-Compiler upon setting the S:32 and S:64 bits. This causes two rules to trigger the capturing of TCP and UDP flows.

Table 22 shows two rules which trigger the capturing of TCP and UDP flows.

Table 22 Capturing TCP and UDP Flows

alert on c0 tcp any any -> any any (msg:"Z TCP within was issued previously for this flow = capture flow"; S:32; R:2; C:32;)
alert on c0 udp any any -> any any (msg:"Z UDP within was issued previously for this stream = capture stream"; S:64; R:2; C:64;)

Anomalous TCP Flags

Some TCP packets with anomalous flags are captured by default to provide scan detection software diagnosis information. Table 23 shows rules which were derived from the Snort scan pre-processor.

Table 23 TCP Packets with Anomalous Flags

alert on c0 tcp any any -> any any (msg:"Z SAPU TCP Flags"; flags:SAPU;)
alert on c0 tcp any any -> any any (msg:"Z FU TCP Flags"; flags:FU;)
alert on c0 tcp any any -> any any (msg:"Z PF TCP Flags"; flags:PF;)
alert on c0 tcp any any -> any any (msg:"Z UP TCP Flags"; flags:UP;)
alert on c0 tcp any any -> any any (msg:"Z Zero TCP Flags"; flags:0;)

The compiler also automatically produces rules that match all packets that are IP fragments or have IP options. These rules are not specified in the *rules.meta* file as they can be more efficiently implemented by the compiler directly.

Appendix A Basic Unix Commands

Unix Commands

Table 24 Basic Unix Commands

Command	Description
cd <path>	This command changes the current directory to the specified directory. The path specified can be an absolute path, or a relative path: • The absolute path begins with a forward slash, and specifies the destination directory beginning from the top of the directory tree. • The relative path does not begin with a forward slash, and specifies the destination beginning from a point common between the current and destination directories.
grep <text> <filename>	This command searches the specified file for a specified string of characters.
h	This command displays help information.
logout	This command logs you out of the current session.
ls <directory>	This command displays the contents of the specified directory.
man <command>	This command diplays the online manual pages for the specified command.
mkdir <directory>	This command makes a directory in the specified location.
more <filename>	This command displays the contents of a file one screenful at a time.
mvdir <directory> <target>	This command moves the specified directory to the target location.
passwd	This command allow you to change the current password.
pwd	This command displays the directory in which you are currently (present working directory).
rmdir <directory>	This command removes the specifed directory. Two conditions apply to this command: • The specified directory must be empty. • The specified directory must not be between the current directory and root directory.

vi Commands

vi has two modes:

- *Command Mode*: In command mode, commands can be entered which allow you to jump to points in a file, search text, and exit the editor.
- *Insert Mode*: Insert mode allows you to create or alter text in a file.

Note: Commands are case sensitive.

Table 25 Basic *vi* Commands

Command	Description
vi <filename>	This command opens the specified file in the editor. If the filename does not exits, *vi* will create it. Enter this command from the Unix shell prompt.
(Escape Key)	This command exits Insert Mode and enters Command Mode.
(Arrow Keys)	These keys move the cursor up, down, left, and right.
i	This command enters Insert Mode and allows you to insert text at the current cursor position.
x	This command deletes the character at the current cursor position.
{/|?}<text>	This command searches for the specified text in the forward direction (using ***/***) or backward direction (using **?**).
[n|1]G	• The command **nG** moves the cursor to the specified line, where *n* is the line number. • The command **1G** moves the cursor to the first line in the file. • The command **G** moves the cursor to the last line in the file.
0	This command moves the cursor to the beginning of the current line.
$	This command moves the cursor to the end of the current line.
:set {number|no number}	This command turns the line numbers on and off.
:q!	This command exits the editor without saving changes.
:wq!	This command saves changes and exits the editor.

Appendix B Glossary

ACK An Acknowledgment packet (ACK) is a packet that is sent from the client to the server to complete a TCP connection. See SYN.

Bro Bro is an open source network intrusion detection and prevention system that uses rules created with a special syntax to examine and control specified traffic. It is similar to Snort except that Bro is Unix-based.

CRC Cyclic Redundancy Check (CRC) is a method of checking for errors in data that has been transmitted on a communications link. A sending device applies a 16- or 32-bit polynomial to a block of data that is to be transmitted and appends the resulting cyclic redundancy code to the block. The receiving end applies the same polynomial to the data and compares its result with the result appended by the sender. If they agree, the data has been received successfully. If not, the sender can be notified to resend the block of data.

DHCP Dynamic Host Configuration Protocol (DHCP) is a protocol that automatically requests an IP address, subnet mask, and default gateway for a network client.

DMA Direct Memory Access (DMA) is a method by which devices in a hardware system can transfer data without occupying the CPU. In the case of the P-Series, the network interface card can transfer matched packets directly to the host memory by taking control of the PCI bus.

DPI Dynamic Parallel Inspection (DPI) is an engine based on Multiple Instruction Single Data (MISD) hardware architecture that can simultaneously execute thousands of security policies and capture/blocking operations on the same data.

Dynamic Rules Dynamic rules allocate generic registers inside the firmware to allow you to create and modify rules at runtime without changing the firmware.

Flow A flow is a series of packets with the same state. See State.

FPGA Field Programmable Gate Array (FPGA) is a logic device that is re-programmable; it is a counterpart to the Application-Specific Integrated Circuit (ASIC) that cannot be modified once it has been programmed.

Garbage Collection Garbage is data that is no longer necessary; garbage collection is the process of discarding this data to free resources. In the context of the P-Series, garbage is old state or flows.

IDS/IPS Intrusion Detection System/Intrusion Prevention System

MISD Multiple Instruction Single Data (MISD) is a computer architecture that executes many operations simultaneously on one set of data. It is a counterpart to Single Instruction Multiple Data (SIMD) and Multiple Instruction Multiple Data (MIMD) architectures.

Null Firmware Null firmware is firmware that has no static rules. Null firmware is used to maximize the dynamic rule capacity on the FPGA.

Offset Offset is another term for the TCP header length.

Rules.meta

Rules.meta is a Snort rules file supplied with the P-Series appliance by Force10. The rules in this file report on flow information and handle possible TCP segmentation evasion attempts. They also provide compatibility with Snort, and including them allows you to run Snort on the DPI interface.

SFP

Small Form-factor Pluggable (SFP) is an optical transceiver that interfaces a network device and a fiber or unshielded twisted pair (UTP) network cable. SFPs support the SONET and Gigabit Ethernet standards and can transmit data at a rate of 4.25 Gb/s.

Snort

Snort is an open source network intrusion detection and prevention system that uses rules created with a special syntax to examine and control specified traffic.

SPAN Port

Switched Port Analyzer (SPAN) Port is a switch port that receives a copy of specific traffic that passes through a switch. The SPAN port is also called a mirroring port.

State

State is information about a flow including the source address, destination address, source port, and destination port. See Flow.

Static Rules

Static rules are rules that are specified in a file using Snort syntax, and then compiled to become part of the firmware. Static rules can be disabled/enabled individually, but they cannot be changed once they have been loaded into the FPGA. To change static rules, you make changes to the rules in the original rules file, recompile them, and reload the new firmware in the FPGA.

SYN

A synchronous packet (SYN) is a packet sent from the client to the server that requests a TCP connection. It is the first part of the TCP handshake that establishes a TCP connection between the client and server.

The second part of the handshake is where the server sends a SYN-ACK packet back to the client to acknowledge the receipt of the SYN request. Finally, the client sends an ACK packet to the server to complete the connection. A SYN flood is a type of denial of service attack where a series of handshakes is initiated but not completed because the final ACK packet is never sent to the server. This occupies the server’s resources, which results in a denial of service for other clients. See ACK.

Tap

A tap is a device that can passively monitor network traffic, and is analogous to a telephone wire tap.

XPAK

XPAK is a transceiver that interfaces a network device and a fiber or unshielded twisted pair (UTP) network cable. It can transmit data at a rate of 10 Gb/s.