Ethanalyzer: Cisco NX-OS Software Built-In Packet Capture Utility

White Paper

June 08, 2011

What You Will Learn **3**
Packet-Capture Tools **3**
Cisco Nexus NX-OS Implementation **4**
Effect of Ethanalyzer on the CPU **4**
When to Use Ethanalyzer on Cisco NX-OS **4**
How to Configure Live Packet Captures on Cisco NX-OS **5**
How to Read and Export a Capture **6**
Examples **7**
Example 1 7
Example 2 7
Example 3 8
Example 4 9
Example 5 9
Example 6 11
Example 7 11
Conclusion **11**
For More Information **12**
General Public License Considerations **12**

What You Will Learn

Cisco® NX-OS Software is a state-of-the-art operating system for the Cisco Nexus® Family switching platform. Cisco NX-OS is a data center-lass operating system built with modularity, resiliency, and serviceability at its foundation. To help ensure serviceability for mission-critical data center environments, Cisco NX-OS provides comprehensive features, including a built-in protocol analyzer called Ethanalyzer.

Ethanalyzer provides sniffing capabilities to Cisco NX-OS within the operating system, simplifying the need for a third-party network probe to capture control traffic sourced from or destined to the switch, including Spanning Tree Protocol and Link Aggregation Control Protocol (LACP) traffic.

This document explains the various packet capture tools and the Ethanalyzer implementation, based on the open source Terminal Wireshark (TShark) utility and providing additional functions from tcpdump. The document describes the functions available in Ethanalyzer and the circumstances in which they can be used. It also explains how to configure Ethanalyzer to capture and read packet captures in real time. This document illustrates Ethanalyzer configuration with examples.

Packet-Capture Tools

Packet-capture tools, also known as sniffers or packet analyzers, intercept and log traffic that is passing over a network or a part of a network. When data streams flow across the network, the sniffer can capture each packet. If needed, it can decode a packet's raw data, showing the values of the various fields and analyzing the packet's content using RFC and other specifications.

Three of the most popular open source packet capture tools are tcpdump and TShark for UNIX-based systems and Wireshark (previously Ethereal) for Microsoft Windows and Mac OS.

Tcpdump is the most common command-line-based packet analyzer tool. It is distributed under the free Berkeley Software Distribution (BSD) license. It operates on most UNIX-based operating systems: BSD, Linux, Solaris, Mac OS, etc. Tcpdump uses the libpcap library to capture packets. It can capture packet data from a live network or read packets from a previously saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file.

Wireshark is very similar to Tcpdump, with the addition of an open source GUI and visual tools for information sorting and filtering. Wireshark allows the user to monitor the network interfaces, wired or wireless. On Linux, BSD, and Mac OS X, Wireshark uses libpcap; and for Microsoft Windows, it uses the WinPcap library.

Terminal-based TShark is the command-line network protocol analyzer of Wireshark. In addition to tcpdump's functions, it displays a running count of captured packets while the capture is in progress. TShark's native capture file format is also the libpcap format.

Libpcap is a portable C/C++ UNIX library for packet capture (pcap). It consists of an API for capturing network traffic.

Cisco Nexus NX-OS Implementation

Cisco NX-OS is a modular operating system running a Linux Montavista kernel, Version 2.6.10. This implementation facilitates addition of embedded Linux tools to the environment. Cisco implemented Ethanalyzer on Cisco NX-OS, a wrapper over TShark. TShark, as described in the previous section uses libpcap. This implementation enables Ethanalyzer to capture and decode packets using TShark and the libpcap library.

The implementation of Ethanalyzer directly in the Cisco NX-OS command line makes the utility very easy to use, without the need for any additional steps, such as use of a dedicated shell mode.

Effect of Ethanalyzer on the CPU

Since Ethanalyzer is part of the software running on the supervisor, understanding its effect to the supervisor's CPU is important. Testing has shown an average increase in the supervisor's CPU utilization of just under 5 percent. Utilization can be decreased by 1 or 2 percent by saving the capture data in a file using the write option, described later in this document.

When to Use Ethanalyzer on Cisco NX-OS

Ethanalyzer is useful when troubleshooting problems related to the switch itself. The packets captured by Ethanalyzer need to be generated or destined for the switch supervisor CPU itself.

Ethanalyzer offers these main benefits:

- Ethanalyzer, as part of the Cisco NX-OS integrated management tool set, improves troubleshooting and reduces time to resolution.
- Ethanalyzer helps preserve and improve the operational continuity of the network infrastructure.
- Ethanalyzer achieves its benefits by allowing network administrators to learn about the amount and nature of the control-plane traffic within their switches.

Ethanalyzer enables you to accomplish the following functions:

- Capture packets sent to and received by the switch supervisor CPU
- Set the number of packets to be captured
- Set the length of the packets to be captured
- Display packets with either very detailed protocol information or a one-line summary
- Open and save the packet data captured
- Filter packet captures based on many criteria
- Filter packet displays based on many criteria
- Decode the internal header of the control packet

Ethanalyzer does not capture hardware switched traffic between data ports of the switch. For this type of packet capture, you can use Cisco Switch Protocol Analyzer (SPAN). For more information about SPAN, please refer to the related white papers and configuration guides.

How to Configure Live Packet Captures on Cisco NX-OS

First decide what is the traffic of interest to be captured and where it is expected on the switch: the inband ports or the out-of-band management mgmt0 port (Figure 1).

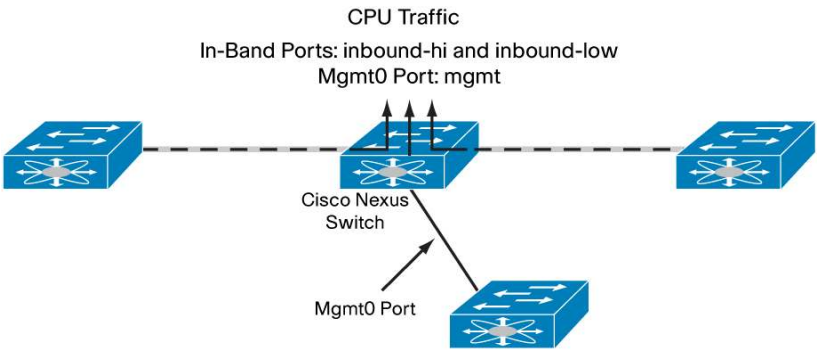
The Cisco Nexus 5000 and 3000 Series Switches have three types of Ethernet interfaces used by Ethanalyzer:

- Eth0 is the mgmt0 port of the switch. It is defined as **mgmt** in Ethanalyzer.
- Eth3 is an inband interface handling traffic from the data ports of the switch. It handles low-priority control packets destined for the switch CPU such as Internet Group Management Protocol (IGMP), TCP, User Datagram Protocol (UDP), IP, and Address Resolution Protocol (ARP) traffic. It is defined as **inbound-low** in Ethanalyzer.
- Eth4 is also an inband interface handing traffic from the data ports of the switch. It handles high-priority control packets destined for the switch CPU, such as Spanning Tree Protocol, Link Aggregation Control Protocol (LACP), Cisco Discovery Protocol, Data Center Bridging Exchange (DCBX), Fibre Channel, and Fibre Channel over Ethernet (FCoE) traffic. It is defined as **inbound-hi** in Ethanalyzer.

The Cisco Nexus 7000 Series has two types of Ethernet interfaces used by Ethanalyzer:

- Eth1 is the mgmt0 port of the switch. It is defined as **mgmt** in Ethanalyzer.
- Eth0 is an inband interface handling traffic from the data ports of the switch. It handles all control packets destined for the switch CPU such as IGMP, TCP, UDP, IP, ARP, Spanning Tree Protocol, LACP, Cisco Discovery Protocol, DCBX, Fibre Channel, and FCoE . It is defined as **inband** in Ethanalyzer.

Figure 1. Ethanalyzer Port Types



Note: All in-band Ethernet ports sending or receiving data over the switch supervisor will be captured with the inband-hi or inband-low option. However, display or capture filtering can be applied.

To start a packet capture with Ethanalyzer, use these commands:

Switch# **ethanalyzer local interface [inbound-hi|inbound-low|mgmt] (options)**

Table 1 lists the options.

Table 1. Ethanalyzer Packet Capture Options

Option	Description
autostop	Capture autostop condition
capture-filter	Filter on Ethanalyzer capture
capture-ring-buffer	Capture ring buffer option
decode-internal	Include internal system header decoding
detail	Display detailed protocol information
display-filter	Display filter on frames captured
limit-captured-frames	Maximum number of frames to be captured (default is 10)
limit-frame-size	Capture only a subset of a frame
write	Filename to which to save capture

By default, Ethanalyzer captures up to 10 frames. Use the **limit-captured-frames** option to change the value; to remove the limit, set the value to 0.

There are two filtering approaches for configuring a packet capture: you can use the display filter, or you can use the capture filter. Both filtering methods can be used when displaying or saving a capture to a file. You can also capture in full detail all the traffic and then export the capture file to the Wireshark utility and filter it in the Wireshark user interface.

Further information about the display and capture filters can be found at:

- http://wiki.wireshark.org/DisplayFilters
- http://wiki.wireshark.org/CaptureFilters

How to Read and Export a Capture

A packet capture file can be read directly on the Cisco NX-OS command line, or the file can be exported.

To locally open a capture file, use this command:

Switch# **ethanalyzer local read [bootflash|usb1|volatile]:filename**

To export the packet capture file, use the copy command:

Switch# **copy bootflash:capture.pcap [destination]:**

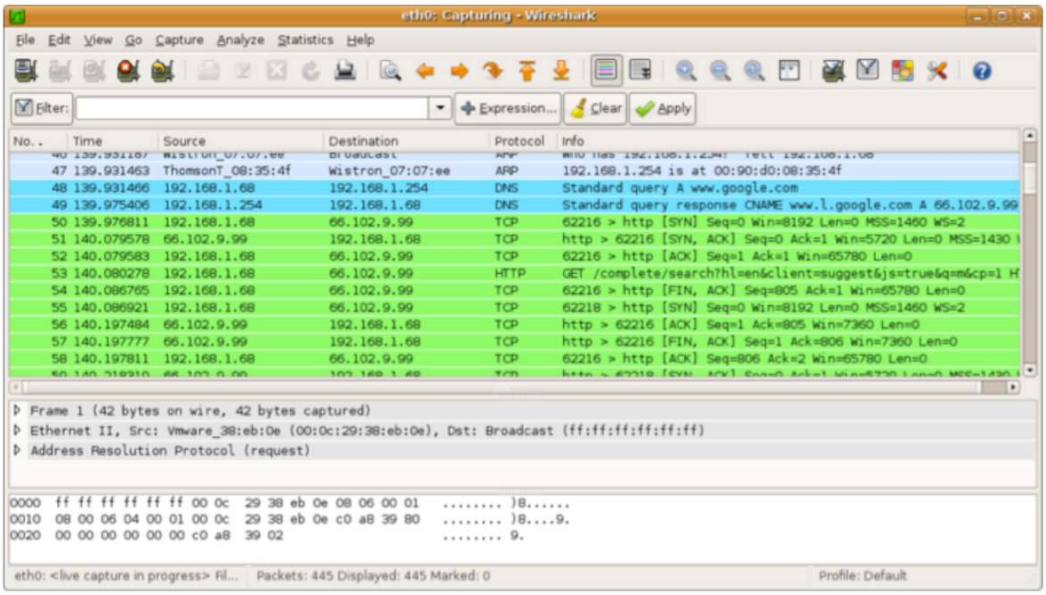
The destination can be any of the options shown in Table 2.

Table 2. Destination Options

Option	Description
ftp:	Specify the destination file system
scp:	Specify the destination file system
sftp:	Specify the destination file system
tftp:	Specify the destination file system
usb1:	Specify the destination file system

After it has been exported, the capture file can be opened with Wireshark to allow easier analysis of the capture, as shown Figure 2.

Figure 2. Wireshark GNU Utility



Examples

Example 1

Capture an unlimited detailed number of packets with decoded internal information and save the capture to a file called capture.pcap:

switch# **ethanalyzer local interface inbound-hi decode-internal detail limit-captured-frames 0 write bootflash:capture.pcap**

Capturing on eth4

13494

Note: Ctrl+C can be use to stop the capture.

Example 2

Capture a specified maximum number of packets - 4 this example:

switch# **ethanalyzer local interface inbound-hi limit-captured-frames 4**

```
Capturing on eth4
2011-05-15 22:01:24.344267 c8:4c:75:4d:52:f4 -> 01:80:c2:00:00:0e LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0134
2011-05-15 22:01:24.344345 c8:4c:75:4d:52:f5 -> 01:80:c2:00:00:0e LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0134
```

```
2011-05-15 22:01:24.670571 c8:4c:75:5b:25:40 -> 00:05:73:b4:f3:a4 LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0120
2011-05-15 22:01:24.670876 c8:4c:75:5b:25:40 -> 00:05:73:b4:f3:a4 LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0120
```

4 packets captured

Example 3

Capture packets not exceeding the specified maximum packet size:

switch# **ethanalyzer local interface inbound-hi limit-frame-size 128**

```
Capturing on eth4
2011-05-15 22:15:59.872401 00:05:73:b4:f3:c6 -> 01:00:0c:cc:cc:cd STP RST. Root =
32770/00:05:73:b4:f3:fc Cost
= 0 Port = 0x809f
2011-05-15 22:16:01.199747 00:0d:ec:f1:3f:db -> 01:80:c2:00:00:00 STP RST. Root =
32769/00:05:73:b4:f3:fc Cost
= 2 Port = 0x9001
2011-05-15 22:16:01.412162 00:05:73:b4:f3:a4 -> c8:4c:75:5b:25:40 LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0120
2011-05-15 22:16:01.412361 c8:4c:75:5b:25:40 -> 00:05:73:b4:f3:a4 LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0120
2011-05-15 22:16:01.412468 00:05:73:b4:f3:a4 -> c8:4c:75:5b:25:40 LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0120
2011-05-15 22:16:01.412623 c8:4c:75:5b:25:40 -> 00:05:73:b4:f3:a4 LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0120
2011-05-15 22:16:01.412749 00:05:73:b4:f3:a4 -> 68:ef:bd:62:28:80 LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0120
2011-05-15 22:16:01.412872 c8:4c:75:5b:25:40 -> 00:05:73:b4:f3:a4 LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0120
2011-05-15 22:16:01.412940 00:05:73:b4:f3:a4 -> c8:4c:75:5b:25:40 LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0120
2011-05-15 22:16:01.412948 68:ef:bd:62:28:80 -> 00:05:73:b4:f3:a4 LLC U, func=UI;
SNAP, OUI 0x00000C (Cisco), PI
D 0x0120
10 packets captured
Program exited with status 0.
```

Note: By default, the capture is limited to 10 frames. This value can be changed with the option **limit-captured-frames** <0-2147483647>. Capture is stopped after the specified number of frames (0 means no limit).

Example 4

Use the display filter to display only Cisco Discovery Protocol packets:

switch# **ethanalyzer local interface inbound-hi decode-internal display-filter "cdp"**

```
Capturing on eth4
2011-05-15 20:50:08.971379 00:05:73:ce:47:c9 -> 01:00:0c:cc:cc:cc CDP Device ID:
switch2(SSI15040AM0) Port ID: E
thernet1/2
2011-05-15 20:50:08.971392 00:05:73:ce:47:f7 -> 01:00:0c:cc:cc:cc CDP Device ID:
switch2(SSI15040AM0) Port ID: E
thernet1/48
2011-05-15 20:50:10.401985 00:05:73:ce:47:c9 -> 01:00:0c:cc:cc:cc CDP Device ID:
switch2(SSI15040AM0) Port ID: E
thernet1/2
2011-05-15 20:50:10.401997 00:05:73:ce:47:f7 -> 01:00:0c:cc:cc:cc CDP Device ID:
switch2(SSI15040AM0) Port ID: E
thernet1/48
2011-05-15 20:50:15.459076 00:22:55:79:b1:c1 -> 01:00:0c:cc:cc:cc CDP Device ID:
switch3(TBM12347542) Port
 ID: Ethernet8/2
```

Example 5

Capture the full frame detail:

switch# **ethanalyzer local interface inbound-hi detail limit-captured-frames 1**

```
Capturing on eth4
Frame 1 (68 bytes on wire, 68 bytes captured)
    Arrival Time: May 15, 2011 22:01:11.892278000
    [Time delta from previous captured frame: 1305496871.892278000 seconds]
    [Time delta from previous displayed frame: 1305496871.892278000 seconds]
    [Time since reference or first frame: 1305496871.892278000 seconds]
    Frame Number: 1
    Frame Length: 68 bytes
    Capture Length: 68 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:vlan:llc:stp]
Ethernet II, Src: 00:05:73:b4:f3:c6 (00:05:73:b4:f3:c6), Dst: 01:00:0c:cc:cc:cd
(01:00:0c:cc:cc:cd)
    Destination: 01:00:0c:cc:cc:cd (01:00:0c:cc:cc:cd)
        Address: 01:00:0c:cc:cc:cd (01:00:0c:cc:cc:cd)
        .... ...1 .... .... .... .... = IG bit: Group address
(multicast/broadcast)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory
default)
    Source: 00:05:73:b4:f3:c6 (00:05:73:b4:f3:c6)
        Address: 00:05:73:b4:f3:c6 (00:05:73:b4:f3:c6)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory
default)
    Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN
    111. .... .... .... = Priority: 7
```

```
    ...0 .... .... .... = CFI: 0
    .... 0000 0000 0001 = ID: 1
    Length: 50
Logical-Link Control
    DSAP: SNAP (0xaa)
    IG Bit: Individual
    SSAP: SNAP (0xaa)
    CR Bit: Command
    Control field: U, func=UI (0x03)
        000. 00.. = Command: Unnumbered Information (0x00)
        .... ..11 = Frame type: Unnumbered frame (0x03)
    Organization Code: Cisco (0x00000c)
    PID: PVSTP+ (0x010b)
Spanning Tree Protocol
    Protocol Identifier: Spanning Tree Protocol (0x0000)
    Protocol Version Identifier: Rapid Spanning Tree (2)
    BPDU Type: Rapid/Multiple Spanning Tree (0x02)
    BPDU flags: 0x3c (Forwarding, Learning, Port Role: Designated)
        0... .... = Topology Change Acknowledgment: No
        .0.. .... = Agreement: No
        ..1. .... = Forwarding: Yes
        ...1 .... = Learning: Yes
        .... 11.. = Port Role: Designated (3)
        .... ..0. = Proposal: No
        .... ...0 = Topology Change: No
    Root Identifier: 32770 / 00:05:73:b4:f3:fc
    Root Path Cost: 0
    Bridge Identifier: 32770 / 00:05:73:b4:f3:fc
    Port identifier: 0x809f
    Message Age: 0
    Max Age: 20
    Hello Time: 2
    Forward Delay: 15
    Version 1 Length: 0
1 packets captured
Program exited with status 0.
```

Example 6

Capture SNMP traffic on the mgmt0 interface:

switch# **ethanalyzer local interface mgmt capture-filter "udp port 161"**

```
Capturing on eth0
2011-05-15 22:28:03.537627 10.19.69.86 -> 10.29.176.91 SNMP get-next-request
2011-05-15 22:28:03.539306 10.29.176.91 -> 10.19.69.86 SNMP get-response
2011-05-15 22:28:03.560820 10.19.69.86 -> 10.29.176.91 SNMP get-next-request
2011-05-15 22:28:03.561745 10.29.176.91 -> 10.19.69.86 SNMP get-response
2011-05-15 22:28:03.580528 10.19.69.86 -> 10.29.176.91 SNMP get-next-request
```

```
2011-05-15 22:28:03.581439 10.29.176.91 -> 10.19.69.86 SNMP get-response
2011-05-15 22:28:03.601476 10.19.69.86 -> 10.29.176.91 SNMP get-next-request
2011-05-15 22:28:03.603037 10.29.176.91 -> 10.19.69.86 SNMP get-response
2011-05-15 22:28:03.622065 10.19.69.86 -> 10.29.176.91 SNMP get-next-request
2011-05-15 22:28:03.623625 10.29.176.91 -> 10.19.69.86 SNMP get-response
10 packets captured
Program exited with status 0.
```

Example 7

Capture ARP traffic on inband ports:

switch# **ethanalyzer local interface inbound-low display-filter "arp"**

```
Capturing on eth3
2011-05-19 07:37:40.953901 00:05:73:d3:a5:3c -> ff:ff:ff:ff:ff:ff ARP Gratuitous
ARP for 1.1.1.1 (Request)
2011-05-19 07:37:42.972280 00:05:73:d3:a5:3c -> ff:ff:ff:ff:ff:ff ARP Gratuitous
ARP for 1.1.1.1 (Request)
2011-05-19 07:37:47.091695 00:05:73:b4:f3:fc -> ff:ff:ff:ff:ff:ff ARP Who has
1.1.1.1? Tell 1.1.1.2
2011-05-19 07:37:47.092539 00:05:73:d3:a5:3c -> 00:05:73:b4:f3:fc ARP 1.1.1.1 is
at 00:05:73:d3:a5:3c
2011-05-19 07:37:47.318960 00:05:73:b4:f3:fc -> ff:ff:ff:ff:ff:ff ARP Gratuitous
ARP for 1.1.1.2 (Request)
```

Conclusion

Network administrators have difficulty gaining complete knowledge of the control-plane traffic. Visibility into this critical component of every network environment crucial to attaining increased control over the network. Cisco NX-OS helps ensure serviceability for mission-critical data center networks by providing a comprehensive set of features, including a built-in protocol analyzer: Ethanalyzer. Ethanalyzer is based on the popular open source Wireshark protocol analyzer, which provides additional features from tcpdump. Ethanalyzer provides a simple tool for analyzing the network traffic destined to and generated by the supervisor and is directly accessible from the Cisco NX-OS CLI, without the need for additional steps and with very limited impingement on the switch CPU.

For More Information

Refer to the documentation (white papers, configuration guides, troubleshooting guides, etc.) at:

- http://www.cisco.com/go/nexus5000
- http://www.cisco.com/go/nexus3000
- http://www.cisco.com/go/nexus2000

General Public License Considerations

The copyrights to certain works contained in Cisco NX-OS Software are owned by third parties and used and distributed under license. Certain components of this software are licensed under GNU General Public License (GPL) Version 2.0 or GNU Lesser General Public License (LGPL) Version 2.1. A copy of each license is available at:

- http://www.opensource.org/licenses/gpl-2.0.php
- http://www.opensource.org/licenses/lgpl-2.1.php

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**

Printed in USA C11-673817-00 06/11