4 Enterprise Network Instrumentation

Accessing traffic can be more difficult than interpreting it. Enterprise networks are often built for performance, not for visibility. When carrying packets is more important than analyzing them, security staff must find ways to capture network traces. After a brief introduction to common methods of gathering traffic, this chapter examines several creative ways to approach the packet collection problem.[1]

Common Packet Capture Methods

Consider the enterprise shown in Figure 4–1. It consists of a perimeter, DMZ, wireless LAN (WLAN), and intranet. All traffic is passed using Category 5e network cables, with two exceptions. Wireless clients use 802.11b and 802.11g media to speak to the wireless access point, and the perimeter router connects to the ISP router with a serial line. This diagram is a simplified network, so you can assume there may be more than three hosts in the DMZ, two wireless clients, and two workstations per access switch.

1. Chapter 3 of *The Tao of Network Security Monitoring: Beyond Intrusion Detection* thoroughly discusses using hubs, SPAN ports, taps, and inline devices to collect packets on wired and wireless links, where appropriate. Readers looking for an introduction to the traffic capture problem, including a discussion of threat models and monitoring zones, should refer to that chapter before reading this one.

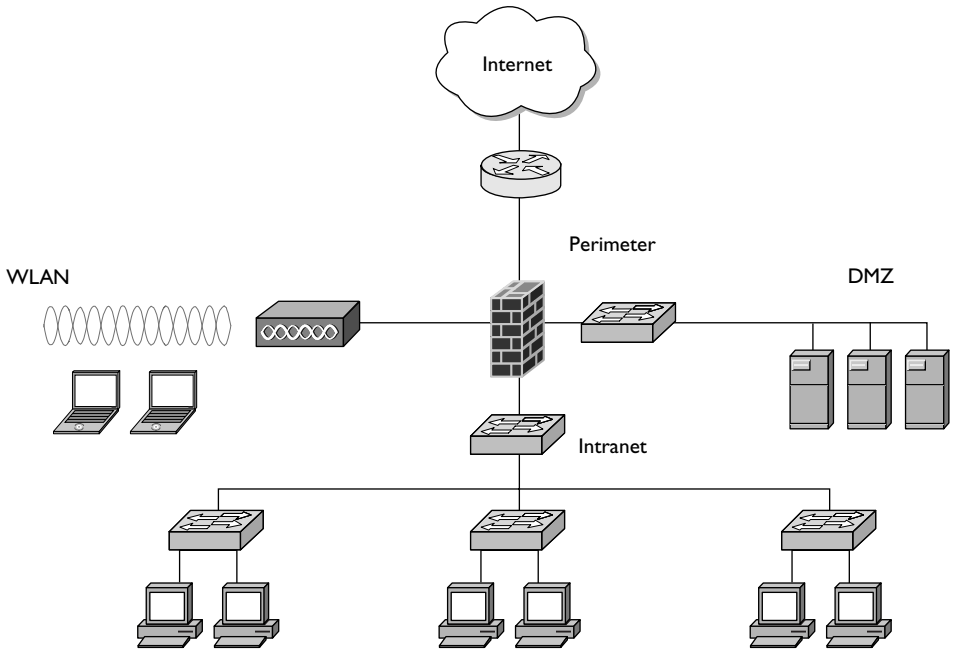


Figure 4–1 Sample Enterprise Network

Accessing traffic in each enterprise zone, with the exception of the WLAN, can be accomplished using one or more of the following standard methods:

- **Hubs.** A hub is a half-duplex networking device that repeats a packet on every interface except the interface that transmitted the packet. All hosts connected to the hub see each other's traffic. Hubs are the low-end means of capturing packets, because they introduce collisions through half-duplex operation. They are also frequently not engineered to meet reliability and uptime requirements found in enterprise-class hardware.
- **SPAN ports.** SPAN stands for "Switched Port Analyzer" and is also referred to as "port mirroring" and "port monitoring." A SPAN port is a port designated on an enterprise-class switch to mirror traffic received on other ports. SPAN ports are a popular packet collection tool because they preserve full duplex links, but there are often not enough SPAN ports available to fulfill every traffic capture need.
- **Taps.** A tap, or test access port, is a networking device specifically designed for monitoring applications. Network taps are used to create permanent access ports for passive

monitoring. Taps sit between any two network devices, such as a router or firewall, two enterprise switches, or a host and an access switch. Taps preserve the full duplex nature of modern switched links.

- **Inline devices.** An inline device is a specialized server or hardware device with more flexibility and complexity than a hub, SPAN port, or tap. Although previous traffic collection products also sit "inline," they are not full-fledged computers running general purpose operating systems. Security staff build inline devices to collect or manipulate traffic as it passes through the inline device itself.

Collecting traffic in the WLAN can be accomplished using the following three methods:

- **Active participation.** A sensor near a wireless access point (WAP) that joins an infrastructure mode WLAN has access to all traffic seen by the WAP. If Wired Equivalent Privacy (WEP) or another means of encrypting wireless traffic is employed, the sensor must be configured with the keys to participate in the WLAN. Traffic captured through active participation tends to look like wired Ethernet traffic to the sensor.
- **Passive participation.** Sensors may collect wireless traffic in a completely passive mode. By not joining the WLAN, the sensor sees all of the control and data traffic passed between the WAP and clients. If encryption is used, the sensor will not be able to see packet contents.
- **Monitoring on the WAP.** It may be possible to collect traffic directly on the WAP itself. This is certainly the case if the WAP is an in-house product built with a general-purpose operating system.

The remainder of this chapter introduces several novel ways to collect traffic that are not found in other security texts. I start by discussing a handful of innovative taps manufactured by Net Optics, Inc. (http://www.netoptics.com).[2]

PCI TAP

Traditional taps are hardware devices that occupy space on a network rack shelf or in an equipment closet. Organizations have recently begun to purchase aggregator taps. These devices combine the two full duplex transmit (TX) lines connected to the tap into a single

2. While I use Net Optics as the example vendor, variations of some of these products are produced by competitors.

output stream suitable for connecting to a monitoring platform. For example, Figure 4–2 shows the Net Optics 10/100 Port Aggregator Tap.

In some environments, administrators don't have the room to deploy a device like this. They may want a solution that is built into the sensor platform. For these situations, the Net Optics PCI Port Aggregator Tap will meet their needs. Figure 4–3 shows the Net Optics PCI Port Aggregator Tap.

This tap plugs into a spare 32 bit, 33 MHz Peripheral Component Interconnect (PCI) slot on a server. The tap uses the server only as a power source. It is completely self-contained, and may even be powered by an external power cord for added redundancy. The tap can be deployed in a system that will collect network traffic, or it may be installed in one system while it provides traffic to a second system. Figure 4–4 explains these two deployment options.

The PCI Port Aggregator Tap is a large device, but it fits into standard server boards. Figure 4–5 shows the PCI Port Aggregator Tap compared to a dual port Gigabit Ethernet network interface card (NIC). The tap is at the top of the picture, and both PCI cards are sitting on top of a 1U Dell PowerEdge 750 rackmount server.

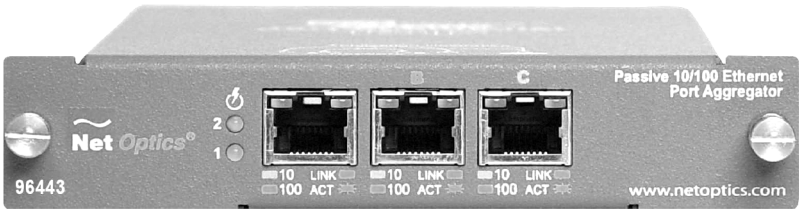


Figure 4–2 Net Optics 10/100 Port Aggregator Tap

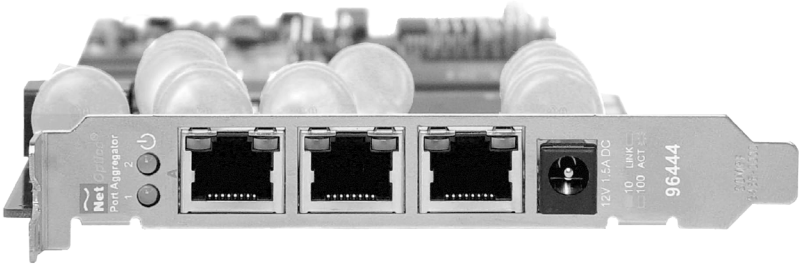
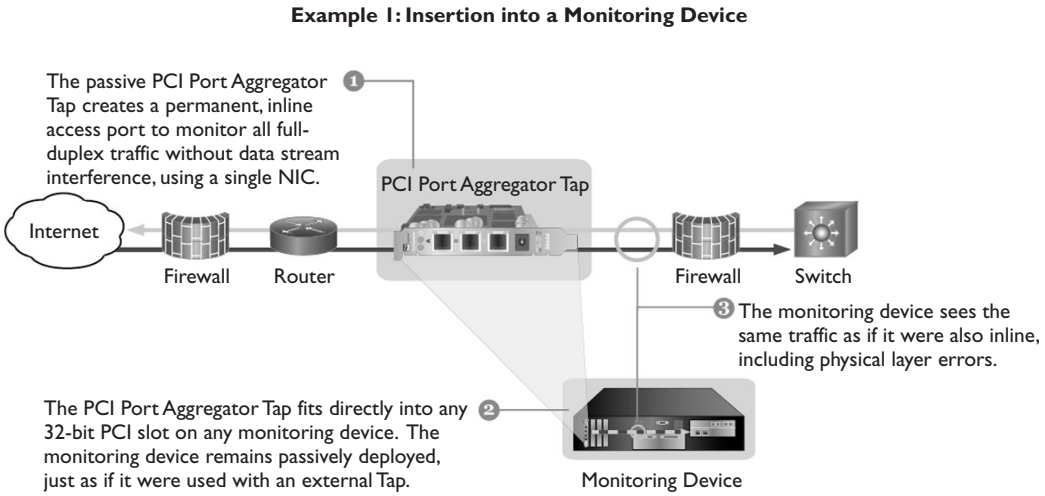


Figure 4–3 Net Optics PCI Port Aggregator Tap



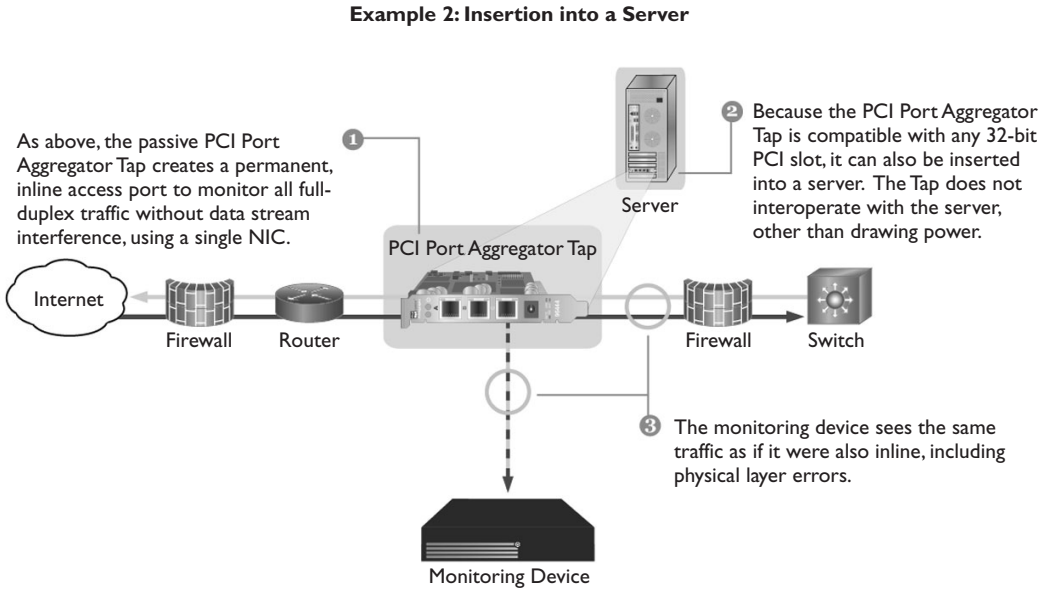


Figure 4–4 PCI Port Aggregator Tap Installation Options

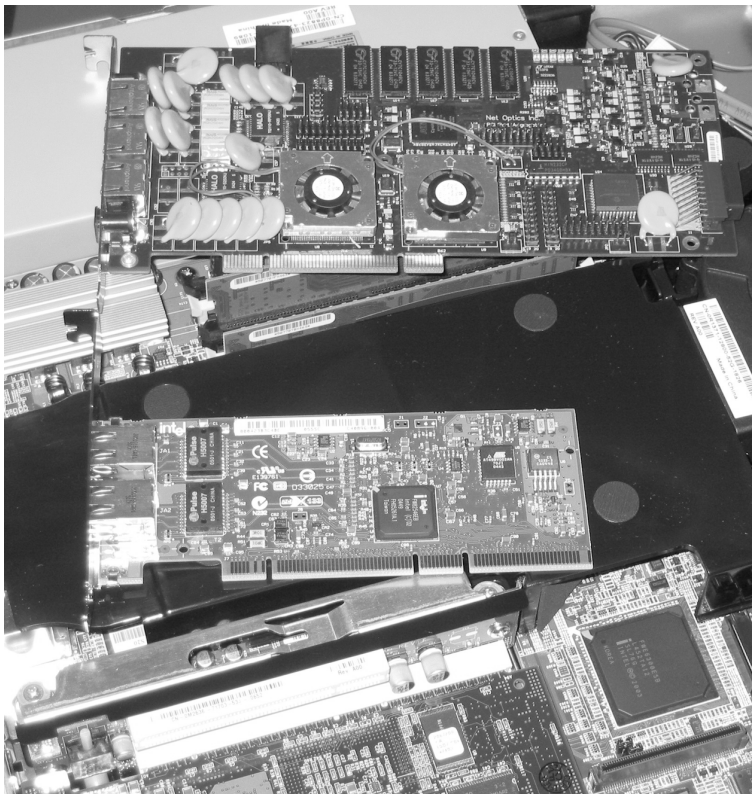


Figure 4–5 PCI Port Aggregator Tap and Dual Gigabit Ethernet NIC. Copyright © 1996–2005 Net Optics, Inc.

The PCI Port Aggregator Tap does not appear to the host operating system as a new network interface. Rather, the tap functions exactly like its external counterpart shown in Figure 4–2. This means the administrator must connect the tap's monitoring port to an interface on the sensor designated to collect network traffic. Figure 4–6 shows a testing scenario where the ports on the PCI Port Aggregator Tap have been cabled to collect and pass network traffic.

The Dell server has two Gigabit Ethernet NICs, seen as em0 and em1 by the FreeBSD operating system. Interface em0 is the top interface on the left side of Figure 4–6. Interface em0 is live and bears an IP address for communicating with the outside world. Interface em1 sits below em0; it is passive and will be used to record network traffic.

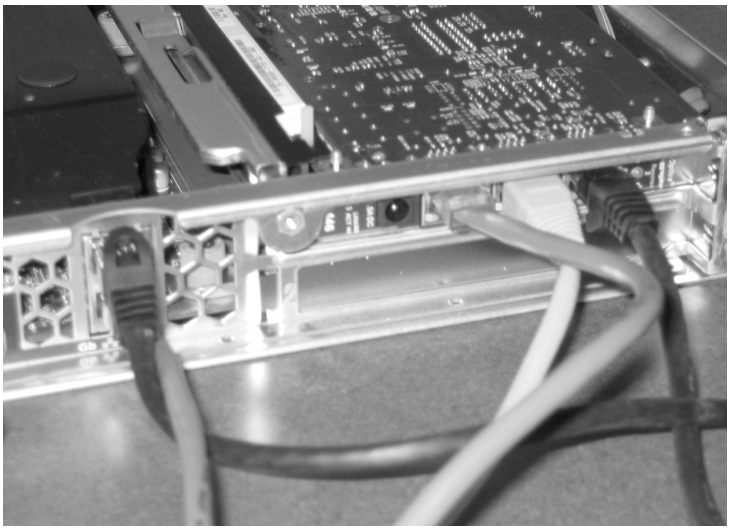


Figure 4–6 PCI Port Aggregator Tap Installed and Cabled

The PCI Port Aggregator tap has three interfaces. The two left-most interfaces are used to passively pass traffic through the tap. The far-right interface transmits copies of the traffic passed through the tap.

In this test deployment, we are essentially watching traffic to and from the server em0 interface by sending copies of that traffic through the tap, to the server em1 interface.

In Figure 4–6, the tap ports are occupied by blue, yellow, and black cables. The yellow cable is connected to a switch with Internet access. The black cable next to the yellow line is connected to the live (IP addressed) interface on the server, em0. The blue cable connects the monitoring interface of the tap to the monitoring interface of the server, em1.

For test purposes, we can send and receive traffic on em0 and watch that traffic on em1. First, I set up em1 to listen to what the tap sends it.

```
sensor# ifconfig em1 -arp up
sensor# tcpdump -n -i em1
tcpdump: WARNING: em1: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full
 protocol decode
listening on em1, link-type EN10MB (Ethernet), capture size
 96 bytes
```

Now I send traffic out em0.

```
$ ping -c 1 www.taosecurity.com
PING www.taosecurity.com (66.93.110.10): 56 data bytes
64 bytes from 66.93.110.10: icmp_seq=0 ttl=55 time=15.240 ms

--- www.taosecurity.com ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/stddev = 15.240/15.240/15.240/0.000 ms
```

Here is what em1 sees courtesy of the tap.

```
09:43:33.263664 IP 192.168.1.41.58947 > 216.182.1.1.53:
 27655+ A? www.taosecurity.com. (37)
09:43:33.276900 IP 216.182.1.1.53 > 192.168.1.41.58947:
 27655 1/2/2 A 66.93.110.10 (131)
09:43:33.277149 IP 192.168.1.41 > 66.93.110.10: icmp 64:
 echo request seq 0
09:43:33.296513 IP 66.93.110.10 > 192.168.1.41: icmp 64:
 echo reply seq 0
```

In an actual deployment, interface em0 would be the sensor's management interface. It would connect to an access switch that enables remote control of the sensor. Interface em1 would still connect to the tap to record traffic passing through the tap. The two passive tap interfaces would sit between network infrastructure devices such as a perimeter router and firewall. Figure 4–7 explains common deployments of the PCI Port Aggregator Tap.

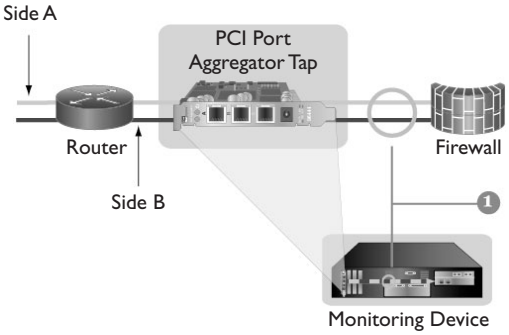
Besides saving space, this tap presents several advantages over common inline devices. Consider building your own passive inline tap using commodity hardware. If any component of the system fails, such as a power supply, NIC, or hard drive, the entire inline device dies. This means traffic will not pass through the inline device and the network goes down.

When using the PCI Port Aggregator Tap, this situation can be avoided. The tap is completely passive. If the server housing the tap fails, the tap will continue to pass traffic. The tap relies on the server only for power, and power is needed only to copy traffic to the tap's monitoring port. The tap is not a single point of failure because it continues to pass traffic even when unpowered. Furthermore, taps are engineered to meet higher customer demands than hubs. Taps also see all layers of network traffic, whereas SPAN ports drop runt or giant (small or large) packets and ignore layer 2 errors.

What if you have more than one sensing system that needs access to network traffic? A solution can be found in the next section.

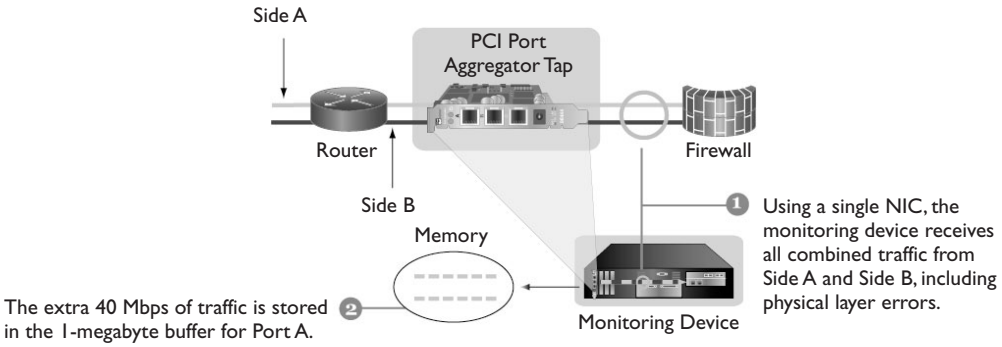
State 1: Side A + Side B is less than or equal to 100% of the NIC's receive capacity

Example: On a 100 Mbps link, Side A is at 30 Mbps and Side B is at 50 Mbps. The NIC receives 80 Mbps of traffic (80% utilization), so no memory is required for the monitoring device NIC to process all full-duplex traffic.



State 2: Side A + Side B becomes greater than 100% of the NIC's receive capacity

Example: There is a burst of traffic, so Side A is now at 90 Mbs while Side B remains at 50 Mbps. The NIC utilization is at 140%, requiring the use of memory to help prevent data loss.



Continued on next page.

Figure 4–7 PCI Port Aggregator Tap Operation. Copyright © 1996–2005 Net Optics, Inc.

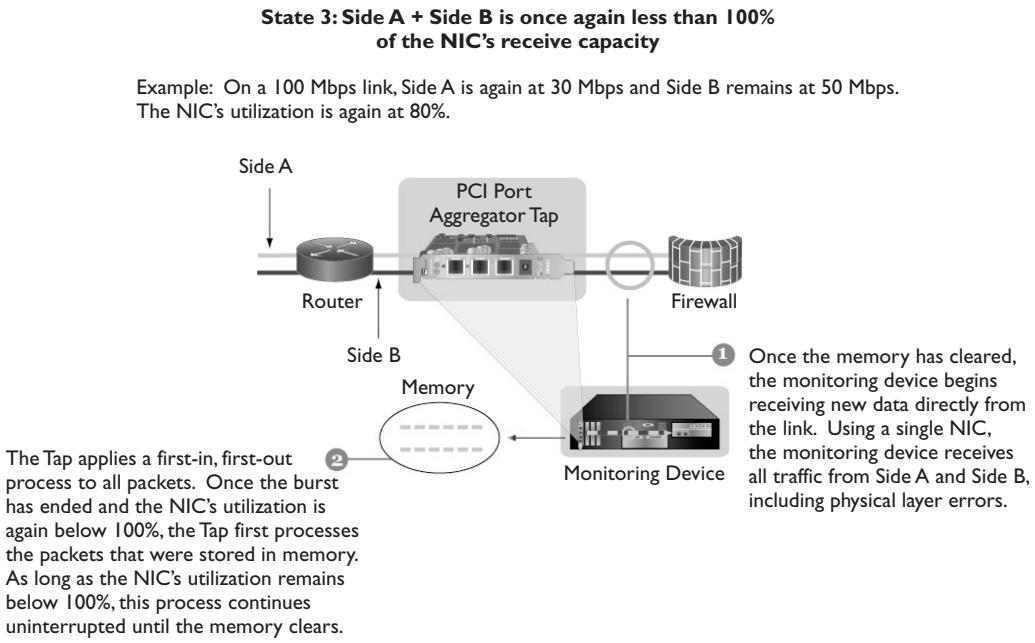


Figure 4–7 *Continued*

DUAL PORT AGGREGATOR TAP

Taps are a great way to collect network traffic, but you may require more than one system to analyze that traffic. For example, you may want your intrusion detection system and your bandwidth monitoring platform to get copies of packets. In situations like these, the Net Optics Dual Port Aggregator may save the day. Figure 4–8 shows the 10/100 Dual Port Aggregator Tap.

This tap is similar to the 10/100 Port Aggregator Tap shown earlier, except it offers two interfaces to which network sensors may connect. The dual tap, like the single tap shown earlier, combines the two transmit (TX) sides of the full duplex traffic passing through the tap into a single full duplex output stream. In a full duplex Ethernet environment, traffic can theoretically be passed at speeds up to 100 Mbps in each direction.

The traffic load dictates where aggregator taps can be deployed. For example, two TX lines, each carrying 40 Mbps, will be aggregated into a single 80 Mbps full duplex stream. The 10/100 aggregator taps will not worry about this load. However, if traffic routinely

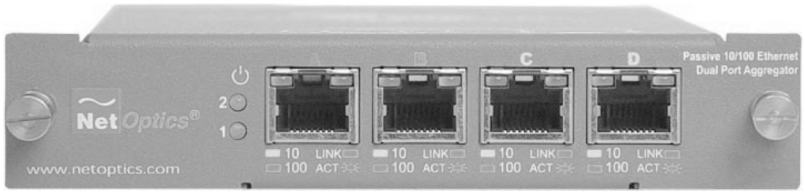


Figure 4–8 Net Optics 10/100 Dual Port Aggregator Tap

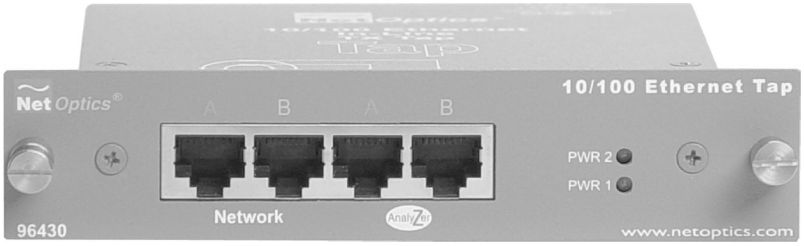


Figure 4–9 Net Optics 10/100 Tap

aggregates above 100 Mbps, say with 60 Mbps on each TX line, the aggregator tap will drop packets once its onboard memory buffer fills.

Because of these concerns, an enterprise running a full duplex link in excess of an aggregated 100 Mbps should use a traditional tap. Traditional taps take each TX line and present them as individual outputs. Figure 4–9 shows a traditional Net Optics 10/100 Tap.

Sensors connected to the traditional tap require a means of bonding the two tap outputs into a single stream. Solutions can be found in *The Tao of Network Security Monitoring: Beyond Intrusion Detection*. This is the form of the script for combining interfaces on FreeBSD as of this writing.[3]

```
#!/bin/sh
# fxp1 and fxp2 are real interfaces which receive tap TX outputs
# ngeth0 is created by ngctl; ng_ether must be loaded so
# netgraph can "see" the real interfaces fxp1 and fxp2
kldload ng_ether
# bring up the real interfaces
```

3. The current version of the script is available at http://www.bejtlich.net/bond.txt.

```
ifconfig fxp1 promisc -arp up
ifconfig fxp2 promisc -arp up
# create ngeth0 and bind fxp1 and fxp2 to it
ngctl mkpeer . eiface hook ether
ngctl mkpeer ngeth0: one2many lower one
ngctl connect fxp1: ngeth0:lower lower many0
ngctl connect fxp2: ngeth0:lower lower many1
ifconfig ngeth0 -arp up
```

Our original question discussed copying traffic to two or more monitoring devices. We saw how a Dual Port Aggregator Tap can passively collect traffic if the aggregated bandwidth of the monitored link does not exceed 100 Mbps. If we want to copy traffic to two or more devices on a link that routinely exceeds an aggregate of 100 Mbps, we should consider using the product in the next section.

2X1 10/100 REGENERATION TAP

Port aggregation for Ethernet networks is limited to situations where the total bandwidth of a full duplex link does not exceed 100 Mbps for 10/100 aggregator taps. Gigabit port aggregator taps face the same problems when total bandwidth exceeds 1000 Mbps. When aggregation is inappropriate, it is best to leave the two TX lines as separate outputs.

Security staff may need to send traffic from the tap to two or more monitoring devices. How can one meet this goal with a traditional tap? The answer lies in the Net Optics 2X1 10/100 Regeneration Tap, shown in Figure 4–10.

On this device, the two ports on the far right connect between network infrastructure elements such as a perimeter router and firewall. The four ports on the left transmit traffic to the sensor. Each pair of ports offers a complete set of network traffic passing through the tap. Figure 4–11 shows how to deploy this tap.

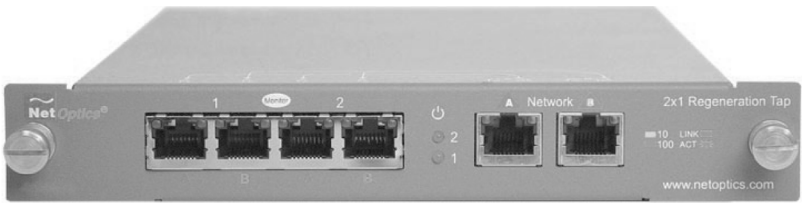


Figure 4–10 Net Optics 2X1 10/100 Regeneration Tap

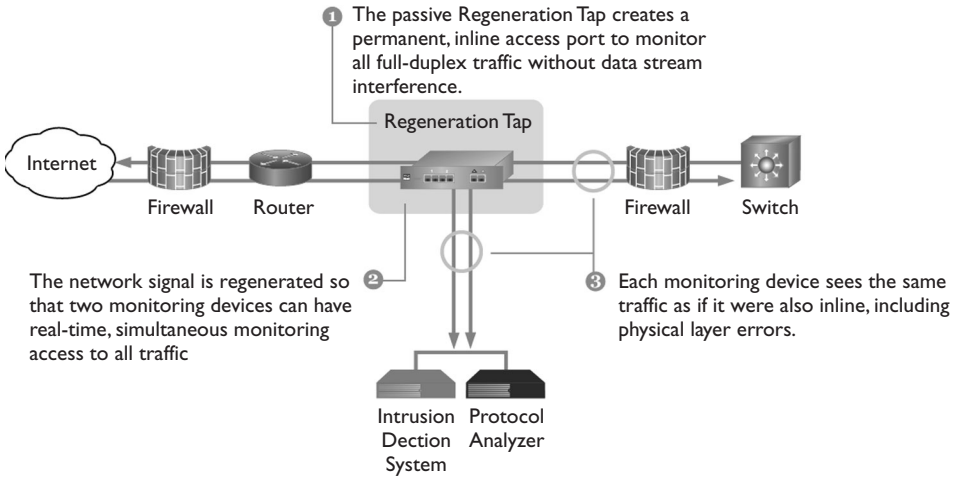


Figure 4–11 2X1 10/100 Regeneration Tap Deployment. Copyright © 1996–2005 Net Optics, Inc.

To understand how this device works, consider the following captures. Here, interfaces sf0 and sf1 sit on one sensor, while sf2 and sf3 are located on a second sensor. Interfaces sf0 and sf2 see "Network A", and sf1 and sf3 connect to "Network B" on the tap. Network A might be traffic leaving the enterprise, while Network B would carry traffic entering the enterprise.

Here is what each interface sees as ICMP traffic passes, as captured and then reviewed with Tcpdump.

```
sensor1# tcpdump -n -r 2x1_regen_tap_pre-bond.sf0.lpc
reading from file 2x1_regen_tap_pre-bond.sf0.lpc,
 link-type EN10MB (Ethernet)
11:14:53.818127 IP 192.168.2.94 > 192.168.2.7: icmp 64:
 echo request seq 0
11:14:54.827103 IP 192.168.2.94 > 192.168.2.7: icmp 64:
 echo request seq 1

sensor1# tcpdump -n -r 2x1_regen_tap_pre-bond.sf1.lpc
reading from file 2x1_regen_tap_pre-bond.sf1.lpc,
link-type EN10MB (Ethernet)
11:14:53.818293 IP 192.168.2.7 > 192.168.2.94: icmp 64:
 echo reply seq 0
11:14:54.827238 IP 192.168.2.7 > 192.168.2.94: icmp 64:
 echo reply seq 1
```

```
sensor2# tcpdump -n -r 2x1_regen_tap_pre-bond.sf2.lpc
reading from file 2x1_regen_tap_pre-bond.sf2.lpc,
 link-type EN10MB (Ethernet)
11:14:53.818082 IP 192.168.2.94 > 192.168.2.7: icmp 64:
 echo request seq 0
11:14:54.827041 IP 192.168.2.94 > 192.168.2.7: icmp 64:
 echo request seq 1

sensor2# tcpdump -n -r 2x1_regen_tap_pre-bond.sf3.lpc
reading from file 2x1_regen_tap_pre-bond.sf3.lpc,
 link-type EN10MB (Ethernet)
11:14:53.818248 IP 192.168.2.7 > 192.168.2.94: icmp 64:
 echo reply seq 0
11:14:54.827186 IP 192.168.2.7 > 192.168.2.94: icmp 64:
 echo reply seq 1
```

This is good, but we would prefer to have interfaces sf0 and sf1 bonded on the first sensor to show a single stream. We also want sf2 and sf3 bonded on the second sensor for the same reason. Once bonded, the traffic looks like this on virtual interfaces ngeth0 and ngeth1.

```
sensor1# tcpdump -n -r 2x1_regen_tap_post-bond.ngeth0.lpc
reading from file 2x1_regen_tap_post-bond.ngeth0.lpc,
 link-type EN10MB (Ethernet)
11:25:01.763213 IP 192.168.2.94 > 192.168.2.7: icmp 64:
 echo request seq 0
11:25:01.763345 IP 192.168.2.7 > 192.168.2.94: icmp 64:
 echo reply seq 0
11:25:02.767909 IP 192.168.2.94 > 192.168.2.7: icmp 64:
 echo request seq 1
11:25:02.767918 IP 192.168.2.7 > 192.168.2.94: icmp 64:
 echo reply seq 1

sensor2# tcpdump -n -r 2x1_regen_tap_post-bond.ngeth1.lpc
reading from file 2x1_regen_tap_post-bond.ngeth1.lpc,
 link-type EN10MB (Ethernet)
11:25:01.763202 IP 192.168.2.94 > 192.168.2.7: icmp 64:
 echo request seq 0
11:25:01.763336 IP 192.168.2.7 > 192.168.2.94: icmp 64:
 echo reply seq 0
11:25:02.767873 IP 192.168.2.94 > 192.168.2.7: icmp 64:
 echo request seq 1
11:25:02.767893 IP 192.168.2.7 > 192.168.2.94: icmp 64:
 echo reply seq 1
```

This 2X1 device isn't the only option. There are also 4X1 and 8X1 versions available, if you need to send traffic to more than two sensors. There are also Gigabit and fiber variations of all of these devices.

Thus far we've looked at taps as independent, passive packet collection systems. We have discussed deploying them in the perimeter, perhaps between the border router and firewall. This situation works for tapping single links, but not for observing traffic between multiple devices on a single switch. SPAN ports are typically the way to gain access to intra-switch traffic, but we have not talked about combining taps with SPAN ports. Is there a way to do so? The answer is yes.

2X1 10/100 SPAN REGENERATION TAP

Some organizations make heavy use of SPAN ports on enterprise switches to capture network traffic. This is the traditional way to observe intra-switch communications, as would be the case for the systems in the DMZ or the workstations in Figure 4–1. A problem arises if one needs to send copies of the SPAN traffic to more than one sensor, and if one wants to send output from two SPAN ports to more than one sensor. That's where the Net Optics 2X1 10/100 SPAN Regeneration Tap comes into play. The device is pictured in Figure 4–12.

The 2X1 10/100 SPAN Regeneration Tap looks nearly identical to the 2X1 10/100 Regeneration Tap shown in Figure 4–10. It is not electrically identical, however. The two interfaces on the far right of the box don't connect to your router and firewall, as they did on the 2X1 10/100 Regeneration Tap. Instead, they connect to SPAN ports from your enterprise class switches. Whatever is connected to the port at the far right is duplicated on the ports labeled "B" on the left side of the tap. Whatever is connected to the port second furthest from the right is duplicated on the "A" ports on the left side of the tap. Furthermore, the two ports on the far right of the 2X1 10/100 SPAN Regeneration Tap cannot pass traffic through themselves, as is the case with the 2X1 10/100 Regeneration Tap.

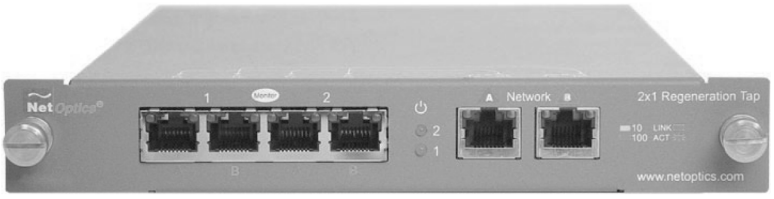


Figure 4–12 2X1 10/100 SPAN Regeneration Tap. Copyright © 1996–2005 Net Optics, Inc.

This is a regeneration tap, meaning it is designed to take copies of observed traffic and send them to more than one sensor. In the following traces, interfaces sf0 and sf1 are again on one sensor, and sf2 and sf3 are on a second sensor.

```
sensor1# tcpdump -n -r span_tap_sf0.lpc
reading from file span_tap_sf0.lpc, link-type EN10MB (Ethernet)
11:47:55.784352 IP 192.168.2.10.56047 > 192.168.2.7.53:
 50548+ A? www.taosecurity.com. (37)
11:47:55.785463 IP 192.168.2.7.53 > 192.168.2.10.56047:
 50548* 1/1/1 A 66.93.110.10 (90)
11:47:55.797978 IP 192.168.2.10.56047 > 192.168.2.7.53:
 50548+ A? www.taosecurity.com. (37)
11:47:55.805704 IP 192.168.2.7.53 > 192.168.2.10.56047:
 50548* 1/1/1 A 66.93.110.10 (90)

sensor1# tcpdump -n -r span_tap_sf1.lpc
reading from file span_tap_sf1.lpc, link-type EN10MB (Ethernet)
11:50:33.465715 IP 192.168.2.10.58009 > 192.168.2.7.53:
 56871+ A? www.sguil.net. (31)
11:50:33.587480 IP 192.168.2.7.53 > 192.168.2.10.58009:
 56871 3/5/4 CNAME wfb.zoneedit.com., A 207.234.129.65,
 A 216.98.141.250 (248)
11:50:33.590831 IP 192.168.2.10.58009 > 192.168.2.7.53:
 56871+ A? www.sguil.net. (31)
11:50:33.687485 IP 192.168.2.7.53 > 192.168.2.10.58009:
 56871 3/5/4 CNAME wfb.zoneedit.com., A 207.234.129.65,
 A 216.98.141.250 (248)

sensor2# tcpdump -n -r span_tap_sf2.lpc
11:47:55.784265 IP 192.168.2.10.56047 > 192.168.2.7.53:
reading from file span_tap_sf2.lpc, link-type EN10MB (Ethernet)
 50548+ A? www.taosecurity.com. (37)
11:47:55.785390 IP 192.168.2.7.53 > 192.168.2.10.56047:
 50548* 1/1/1 A 66.93.110.10 (90)
11:47:55.797888 IP 192.168.2.10.56047 > 192.168.2.7.53:
 50548+ A? www.taosecurity.com. (37)
11:47:55.805617 IP 192.168.2.7.53 > 192.168.2.10.56047:
 50548* 1/1/1 A 66.93.110.10 (90)

sensor2# tcpdump -n -r span_tap_sf3.lpc
reading from file span_tap_sf3.lpc, link-type EN10MB (Ethernet)
11:50:33.465631 IP 192.168.2.10.58009 > 192.168.2.7.53:
 56871+ A? www.sguil.net. (31)
11:50:33.587403 IP 192.168.2.7.53 > 192.168.2.10.58009:
 56871 3/5/4 CNAME wfb.zoneedit.com., A 207.234.129.65,
 A 216.98.141.250 (248)
```

```
11:50:33.590747 IP 192.168.2.10.58009 > 192.168.2.7.53:
 56871+ A? www.sguil.net. (31)
11:50:33.687403 IP 192.168.2.7.53 > 192.168.2.10.58009:
 56871 3/5/4 CNAME wfb.zoneedit.com., A 207.234.129.65,
 A 216.98.141.250 (248)
```

So what are we looking at? It appears sf0 and sf2 (on different sensors) see the same SPAN port output, which here shows a DNS request and reply for www.taosecurity.com. Interfaces sf1 and sf3 (again on different sensors) see SPAN port output from a different switch, where a DNS request and reply for www.sguil.net has been recorded.

There is an important difference between these four traces for interfaces sf0 - sf3 and the four traces shown for sf0 - sf3 for the 2X1 10/100 Regeneration Tap. The 2X1 10/100 Regeneration Tap showed only half-duplex traffic, meaning packets sent in one direction only. We used bonding to bring interfaces sf0 and sf1 on sensor1 together, and sf2 and sf3 on sensor2 together, to display a single full duplex stream on each sensor.

Here, we are already looking at full duplex output on each interface, sf0 - sf3. Remember that we are getting our packets from two separate SPAN ports in this case. The tap is not directly inline—two enterprise switches are collecting traffic and sending it to the tap. There is no need to bond interfaces here because we are already looking at full duplex streams as provided by the switch SPAN ports.

However, we could bond interfaces sf0 and sf1 together on one sensor, and sf2 and sf3 on the other sensor, if we wanted to present a single virtual interface to the sniffing software on each platform. If we do that, we can now see the output from two SPAN ports combined into a single virtual interface. It would look something like this.

```
11:55:06.032533 IP 192.168.2.10.56047 > 192.168.2.7.53:
 50548+ A? www.taosecurity.com. (37)
11:55:06.036645 IP 192.168.2.10.58009 > 192.168.2.7.53:
 56871+ A? www.sguil.net. (31)
11:55:06.037014 IP 192.168.2.7.53 > 192.168.2.10.56047:
 50548* 1/1/1 A 66.93.110.10 (90)
11:55:06.041972 IP 192.168.2.7.53 > 192.168.2.10.58009:
 56871 3/5/4 CNAME wfb.zoneedit.com., A 207.234.129.65,
 A 216.98.141.250 (248)
11:55:06.045319 IP 192.168.2.10.56047 > 192.168.2.7.53:
 50548+ A? www.taosecurity.com. (37)
11:55:06.062005 IP 192.168.2.7.53 > 192.168.2.10.56047:
 50548* 1/1/1 A 66.93.110.10 (90)
11:55:06.065079 IP 192.168.2.10.58009 > 192.168.2.7.53:
 56871+ A? www.sguil.net. (31)
11:55:06.072073 IP 192.168.2.7.53 > 192.168.2.10.58009:
 56871 3/5/4 CNAME wfb.zoneedit.com., A 207.234.129.65,
 A 216.98.141.250 (248)
```

```
11:55:06.075315 IP 192.168.2.10.56047 > 192.168.2.7.53:
 50548+ A? www.taosecurity.com. (37)
11:55:06.081956 IP 192.168.2.7.53 > 192.168.2.10.56047:
 50548* 1/1/1 A 66.93.110.10 (90)
11:55:06.085050 IP 192.168.2.10.58009 > 192.168.2.7.53:
 56871+ A? www.sguil.net. (31)
11:55:06.101978 IP 192.168.2.7.53 > 192.168.2.10.58009:
 56871 3/5/4 CNAME wfb.zoneedit.com., A 207.234.129.65,
 A 216.98.141.250 (248)
```

So how might you use this device in real life? You may have an enterprise switch mirroring traffic to a SPAN port. You would like multiple sensors to watch that SPAN port. Rather than send the traffic from the SPAN port into a cheap hub, you send the SPAN output to a 2X1 (or 4X1 or 8X1) 10/100 SPAN Regeneration tap. Everything stays at full duplex for highest performance. Figure 4–13 demonstrates deploying the 2X1 10/100 SPAN Regeneration Tap.

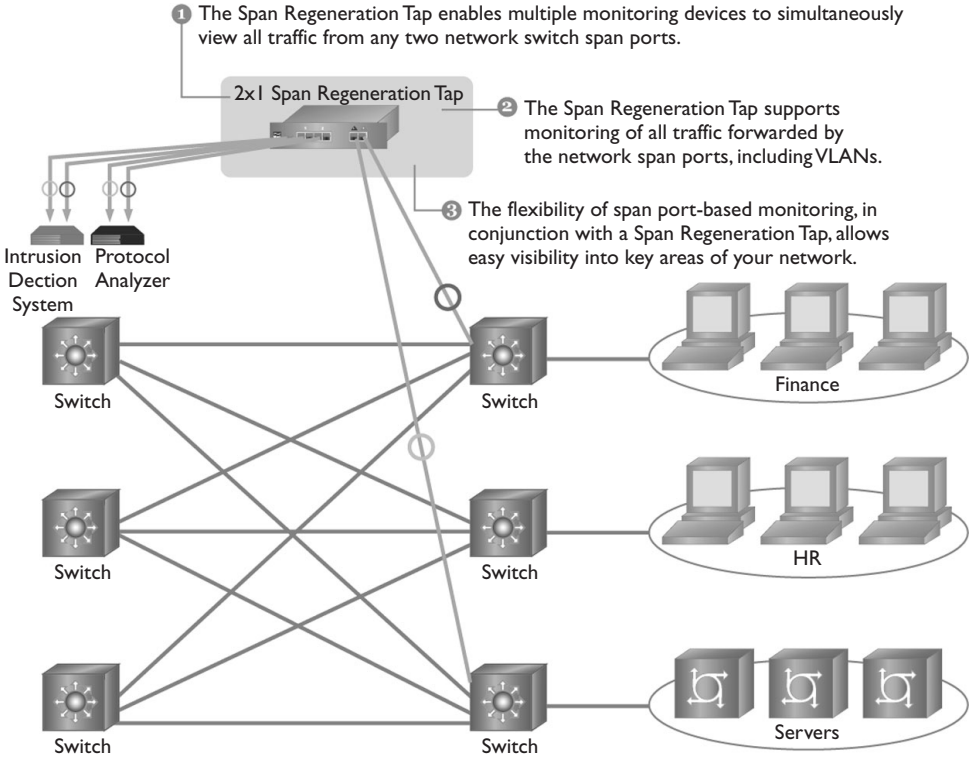


Figure 4–13 Deploying the 2X1 10/100 SPAN Regeneration Tap. Copyright © 1996–2005 Net Optics, Inc.

Hubs are *never* an option for combining traffic from separate TX lines provided by a traditional tap. If one is tapping two SPAN ports, hubs are also *never* an option to provide access for multiple sensors. Consider the consequences of sending tapped traffic to a hub. If traffic enters the hub at the same time, the packets collide but no retransmission occurs. The tap is passive, so those collided packets are lost forever. Taps and hubs never mix!

SPAN ports are a popular way to access intra-switch traffic, but they are not the only way. A new class of devices called matrix switches offer even more flexibility.

MATRIX SWITCH

A **matrix switch** is a device that provides on-demand access to any switch ports requested by the administrator. Matrix switches come in two varieties: inline and SPAN. First, let's discuss inline matrix switches. An administrator may be responsible for numerous enterprise switches. If the administrator wishes to copy traffic from an interface on one of those switches, she must configure a SPAN port on the switch. The SPAN port must be connected to a sensor. If the administrator wants the sensor to have access to SPAN ports on multiple switches, she must connect the SPAN ports on the switches to multiple interfaces on the sensor. A sensor with more than eight interfaces can quickly run out of hardware interrupts, making this a dicey option. Alternatively, she might use another switch to aggregate traffic collected by the first series of switches. If this is starting to sound complicated, you're right—it is!

There is another way. Rather than configuring SPAN ports on access switches, the administrator can choose to deploy an inline matrix switch that complements the enterprise switch. The matrix switch provides network visibility, while the enterprise switch handles moving packets between hosts as it always has. Figure 4–14 shows a deployment scenario for a matrix switch.

When an administrator wishes to see traffic from an interface connected to the inline matrix switch, she remotely configures the matrix switch to monitor the desired interface. The matrix switch begins copying traffic from the designated port to the matrix switch monitoring port, which is connected to a single interface on a sensor. The enterprise switch is completely unaware of any of this activity and continues to forward traffic. The administrator typically controls the matrix switch using a serial-accessible GUI. Figure 4–15 shows the Net Optics 1x16 10/100 In-Line SpyderSwitch, where "SpyderSwitch" is Net Optics' market name for one line of matrix switches.

That explains the inline matrix switch. A SPAN matrix switch offers a second set of options. Imagine the same administrator is also responsible for a few dozen enterprise switches mounted in a set of racks. An inline matrix switch sitting between switches and

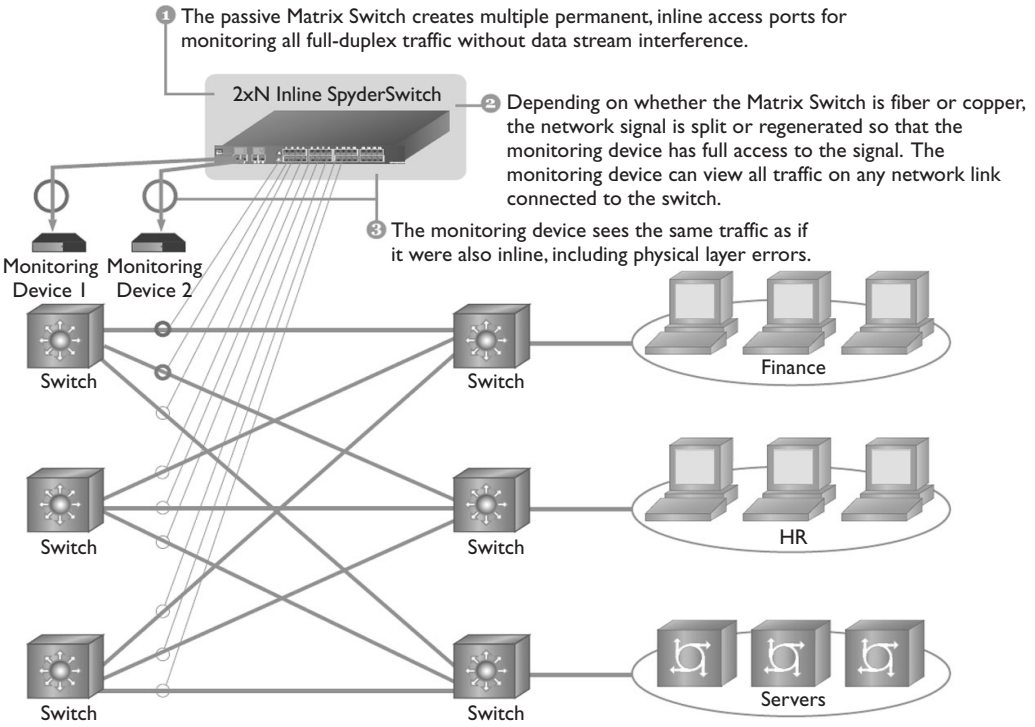


Figure 4–14 Matrix Switch Deployment. Copyright © 1996–2005 Net Optics, Inc.

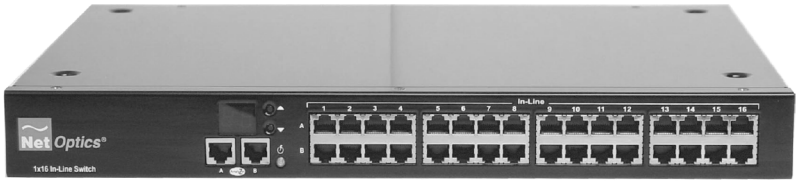


Figure 4–15 Net Optics 1x16 10/100 In-Line SpyderSwitch.

hosts, or between switches and other switches (on trunk lines), may not scale as well. Instead of using inline matrix switches, the administrator can use a SPAN matrix switch. The SPAN matrix switch plugs into SPAN ports on enterprise switches.

When the administrator wants to mirror traffic from an enterprise switch, she configures it to copy packets to its SPAN port. She then tells her SPAN matrix switch to watch

Figure 4–16 Net Optics Link Aggregator Tap

traffic on the interface connected to the SPAN port on the enterprise switch of interest. This system provides an on-demand means of gaining access to hundreds of interfaces on dozens of enterprise switches.

Matrix switches are an excellent choice for high-bandwidth switched networks where on-demand access to network traffic is required. They come in a variety of forms and include versions with multiple outputs, like the 2X1 taps mentioned earlier. If continuous access to a certain number of ports is required, then a different type of aggregator tap might be the solution.

LINK AGGREGATOR TAP

Perhaps your network offers a DMZ like that shown in Figure 4–1. Your DMZ has three or four servers, but your access switch does not provide any SPAN capability. Alternatively, your DMZ access switch may offer a SPAN port, but it is a 10/100 Mbps port like the other ports on the switch. If you want to see all of the traffic passed between hosts in the DMZ, how can you do so and maintain a full duplex environment?

One answer is the matrix switch just shown. Another is the link aggregator tap. This device passively monitors a number of independent network links and aggregates them into a single higher-bandwidth interface. For example, the three DMZ servers connected by 100 Mbps in Figure 4–1 have a theoretical maximum total bandwidth of 600 Mbps. That load could be handled by a single Gigabit interface working at 1000 Mbps. Figure 4–16 shows the Net Optics Link Aggregator Tap, which combines four 10/100 Mbps interfaces into a single Gigabit interface.

A Link Aggregator Tap allows traffic from multiple lower-bandwidth systems to be collected and monitored by a single sensor with a higher-bandwidth NIC. Like other Net Optics products, there are a variety of combinations, such as sending copied traffic to multiple outputs or collecting traffic from fiber and other interface types. We've talked a lot about taps and SPAN ports, so it's time to move to innovative uses of inline devices.

DISTRIBUTED TRAFFIC COLLECTION WITH PF DUP-TO

We've seen network taps that make copies of traffic for use by multiple monitoring systems. These copies are all exactly the same, however. There is no way of using the taps just described to send port 80 TCP traffic to one sensor and all other traffic to another sensor. Commercial solutions like the Top Layer IDS Balancer provide the capability to sit inline and copy traffic to specified output interfaces, based on rules defined by an administrator. Is there a way to perform a similar function using commodity hardware? Of course!

The Pf firewall introduced in Chapter 2 offers the dup-to keyword. This function allows us to take traffic that matches a Pf rule and copy it to a specified interface. Figure 4–17 demonstrates the simplest deployment of this sort of system.

First, we must build a Pf bridge to pass and copy traffic. Here is the `/etc/pf.conf.dup-to` file we will use.

```
int_if="sf0"
ext_if="sf1"
l80_if="sf2"
l80_ad="1.1.1.80"
lot_if="sf3"
lot_ad="2.2.2.2"

pass out on $int_if dup-to ($l80_if $l80_ad) proto tcp from any
 port 80 to any
pass in on $int_if dup-to ($l80_if $l80_ad) proto tcp from any
 to any port 80

pass out on $int_if dup-to ($lot_if $lot_ad) proto tcp from any
 port !=80 to any
pass in on $int_if dup-to ($lot_if $lot_ad) proto tcp from any
 to any port !=80

pass out on $int_if dup-to ($lot_if $lot_ad) proto udp from
 any to any
pass in on $int_if dup-to ($lot_if $lot_ad) proto udp from
 any to any

pass out on $int_if dup-to ($lot_if $lot_ad) proto icmp from
 any to any
pass in on $int_if dup-to ($lot_if $lot_ad) proto icmp from
 any to any
```

To understand this configuration file, we should add some implementation details to our simple Pf dup-to diagram. Figure 4–18 adds those details.

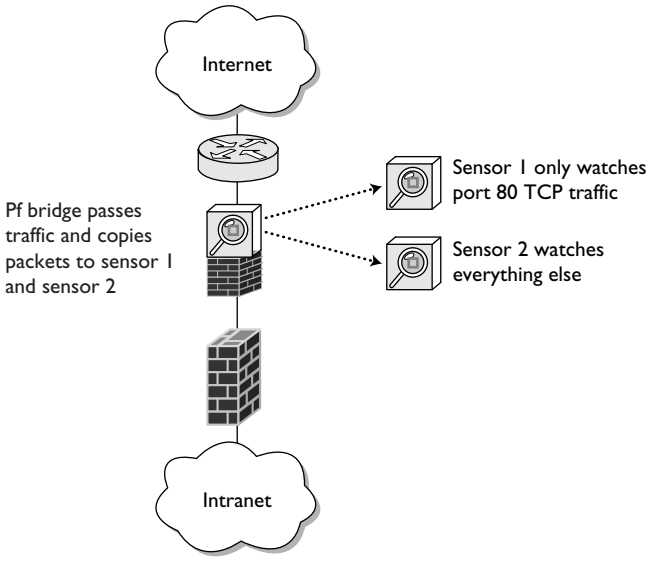


Figure 4–17 Simple Pf Dup-To Deployment

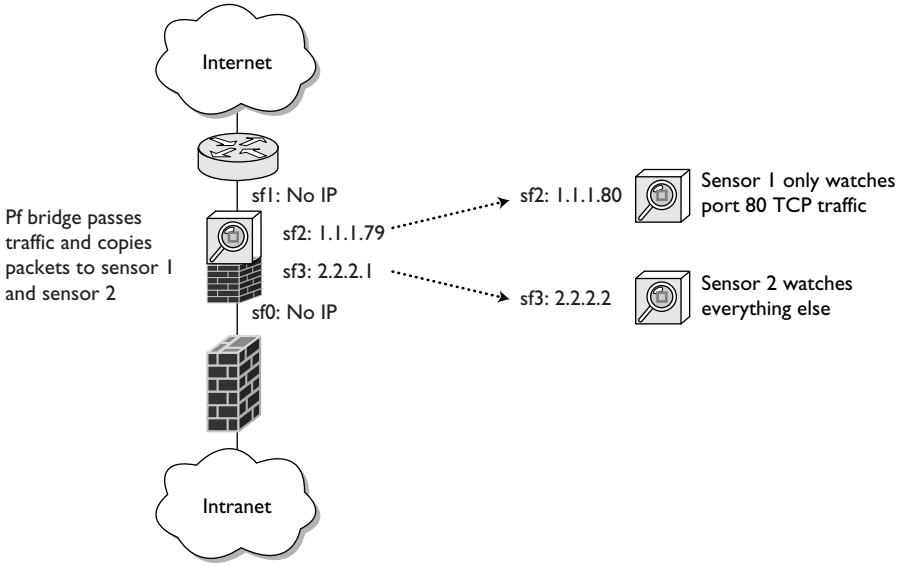


Figure 4–18 Simple Pf Dup-To Implementation Details

To begin, consider the interfaces involved on the Pf bridge:

- Interface sf0 is closest to the intranet. It is completely passive, with no IP address.
- Interface sf1 is closest to the Internet. It is also completely passive, with no IP address.
- Interface sf2 will receive copies of port 80 TCP traffic sent to it by Pf. It bears the arbitrary address 1.1.1.79. Access to this interface by other hosts should be denied by firewall rules, not shown here.
- Interface sf3 will receive copies of all non-port 80 TCP traffic, as well as UDP and ICMP, sent to it by Pf. (For the purposes of this simple deployment, we are not considering other IP protocols.) It bears the arbitrary address 2.2.2.1. Access to this interface by other hosts should be denied by firewall rules, not shown here.

Now consider the two sensors:

- Sensor 1 uses its interface sf2 to capture traffic sent to it from the Pf bridge. It bears the arbitrary IP address 1.1.1.80. Access to this interface by other hosts should be denied by firewall rules, not shown here.
- Sensor 2 uses its interface sf3 to capture traffic sent to it from the Pf bridge. It bears the arbitrary address 2.2.2.2. Access to this interface by other hosts should be denied by firewall rules, not shown here.

One would have hoped the Pf dup-to function could send traffic to directly connected interfaces without the involvement of any IP addresses. Unfortunately, my testing revealed that assigning IP addresses to interfaces on both sides of the link is required. I used OpenBSD 3.7, but future versions may not have this requirement.

With this background, we can begin to understand the `/etc/pf.conf.dup-to` file:

- The first set of declarations define macros for the interfaces and IP addresses used in the scenario.
- The first set of pass commands tells Pf to send port 80 TCP traffic to 1.1.1.80, which is the packet capture interface on sensor 1. Two rules are needed: one for inbound traffic and one for outbound traffic.
- The second set of pass commands tells Pf to send all non-port 80 TCP traffic to 2.2.2.2, which is the packet capture interface on sensor 2. Again two rules are needed.
- The third and fourth set of pass commands sends UDP and ICMP traffic to 2.2.2.2 as well.

Before testing this deployment, ensure Pf is running and that all interfaces are appropriately configured and enabled. To test our distributed collection system, we retrieve the Google home page using `wget`.

```
$ wget http://www.google.com/index.html
--10:19:50--  http://www.google.com/index.html
           => `index.html'
Resolving www.google.com... 64.233.187.99, 64.233.187.104
Connecting to www.google.com[64.233.187.99]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]

    [ <=>                                 ] 1,983       --.--K/s

10:19:51 (8.96 MB/s) - `index.html' saved [1983]
```

Here is what sensor 1 sees on its interface sf2.

```
10:18:58.122543 IP 172.17.17.2.65480 > 64.233.187.99.80:
 S 101608113:101608113(0) win 32768
 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0>
10:18:58.151066 IP 64.233.187.99.80 > 172.17.17.2.65480:
 S 2859013924:2859013924(0) ack 101608114 win 8190 <mss 1460>
10:18:58.151545 IP 172.17.17.2.65480 > 64.233.187.99.80:
 . ack 1 win 33580
10:18:58.153027 IP 172.17.17.2.65480 > 64.233.187.99.80:
 P 1:112(111) ack 1 win 33580
10:18:58.184169 IP 64.233.187.99.80 > 172.17.17.2.65480:
 . ack 112 win 8079
10:18:58.185384 IP 64.233.187.99.80 > 172.17.17.2.65480:
 . ack 112 win 5720
10:18:58.189840 IP 64.233.187.99.80 > 172.17.17.2.65480:
 . 1:1431(1430) ack 112 win 5720
10:18:58.190344 IP 64.233.187.99.80 > 172.17.17.2.65480:
 P 1431:2277(846) ack 112 win 5720
10:18:58.190483 IP 64.233.187.99.80 > 172.17.17.2.65480:
 F 2277:2277(0) ack 112 win 5720
10:18:58.192706 IP 172.17.17.2.65480 > 64.233.187.99.80:
 . ack 2277 win 32734
10:18:58.192958 IP 172.17.17.2.65480 > 64.233.187.99.80:
 . ack 2278 win 32734
10:18:58.204719 IP 172.17.17.2.65480 > 64.233.187.99.80:
 F 112:112(0) ack 2278 win 33580
10:18:58.232685 IP 64.233.187.99.80 > 172.17.17.2.65480:
 . ack 113 win 5720
```

Here is what sensor 2 sees on its interface sf3.

```
10:18:58.089226 IP 172.17.17.2.65364 > 192.168.2.7.53:
 64302+ A? www.google.com. (32)
10:18:58.113853 IP 192.168.2.7.53 > 172.17.17.2.65364:
 64302 3/13/13 CNAME www.l.google.com.,
 A 64.233.187.99, A 64.233.187.104 (503)
```

As we planned, sensor 1 only saw port 80 TCP traffic, while sensor 2 saw everything else. In this case, "everything else" meant a DNS request for www.google.com. So why build a distributed collection system? This section presented a very simple deployment scenario, but you can begin to imagine the possibilities. Network security monitoring (NSM) advocates collecting full content, session, statistical, and alert data. That can be a great amount of strain on a single sensor, even if only full content data is collected.

By building a distributed collection system, NSM data can be forwarded to independent systems specially built for the tasks at hand. In our example, we offloaded heavy Web surfing activity to one sensor and sent all other traffic to a separate sensor.

We also split the traffic-passing function from the traffic-recording function. The Pf bridge in Figure 4–18 is not performing any disk input/output (IO) operations. The kernel is handling packet forwarding, which can be done very quickly. The independent sensors are accepting traffic split out by the Pf bridge. The sensors can be built to perform fast disk IO. This sort of sensor load balancing provides a way to apply additional hardware to difficult packet-collection environments. If adding an inline device running Pf makes you nervous, use a tap instead. Send the tap outputs to the inline device running Pf.

Speaking of difficult environments, critics of NSM are quick to point out that encryption foils capture of full content data. For example, a Web server providing an HTTPS feed over port 443 TCP cannot have its traffic inspected by a network IDS. Or can it?

SQUID SSL TERMINATION REVERSE PROXY

If a Web server offers services via an encrypted channel through port 443 TCP, there is no reason why an attacker should concentrate on exploiting port 80 TCP. Once an intruder encrypts his attack traffic in Secure Sockets Layer (SSL) by communicating with port 443 TCP, a network-based IDS is effectively blinded. It cannot record meaningful full content data, and alert data based on application data will not be generated. While session and statistical data are not affected by encryption, the loss of visibility is a serious issue for security staff.

I described in Chapter 2 how pervasive network awareness requires insight into the traffic passed to each host in the enterprise. Ignoring traffic on port 443 TCP is a common oversight that can have devastating consequences. There are three main options for restoring visibility to encrypted traffic on port 443 TCP, some of which are Web-server specific:

- Deploying a network shim on the Web server that copies traffic from the Web server to a sensor after the Web server decrypts the traffic. Certain commercial products such as Breach SSL (http://www.breach.com) and Ivan Ristic's open source ModSecurity Apache module offer this capability.[4]
- Deploying Apache with mod_proxy in reverse proxy mode.
- Deploying Squid as a reverse proxy.

The first option preserves "end-to-end encryption" to the greatest possible extent, but it relies on deploying additional software on the Web server. The second option requires installing the Apache Web server and is not discussed here; see http://httpd.apache.org/docs-2.0/mod/mod_proxy.html for details. I describe the third option now, because it is the solution that best fits into a network-centric security monitoring scenario.

Let's take a closer look at the term "reverse proxy" by first understanding a forward proxy. A forward proxy sits between a Web client and a Web server. The client is typically configured to talk to the proxy directly when making Web requests, although transparent methods to proxy Web traffic are possible. Upon receiving a Web request, the forward proxy makes the request to the Web server on behalf of the client. This is the same sort of proxy discussed in Chapter 2. Figure 4–19 depicts the use of a forward proxy.

We deployed forward proxies in Chapter 2 to gain visibility and control of the Web traffic leaving our enterprise. Now consider the situation where our enterprise provides a Web server to the outside world. The common implementation appears in Figure 4–20.

If the Web client in Figure 4–20 decides to communicate with our Web server using port 443 TCP, our NSM platform will not be able to inspect the content of the communications. We will not be able to generate useful full content or alert data. However, we can deploy a reverse proxy to change the situation, as depicted in Figure 4–21.

Now, connections to port 443 TCP are terminated on the reverse proxy. The reverse proxy forwards unencrypted connections to port 80 TCP. Our IDS now has visibility if it watches the link between the reverse proxy and the Web server, as shown in Figure 4–21.

Some purists may complain that the "end-to-end encryption" shared by a Web client and server is severed by this arrangement. This is a true statement, but who is benefiting

4. For more information on mod_security, visit http://www.modsecurity.org.

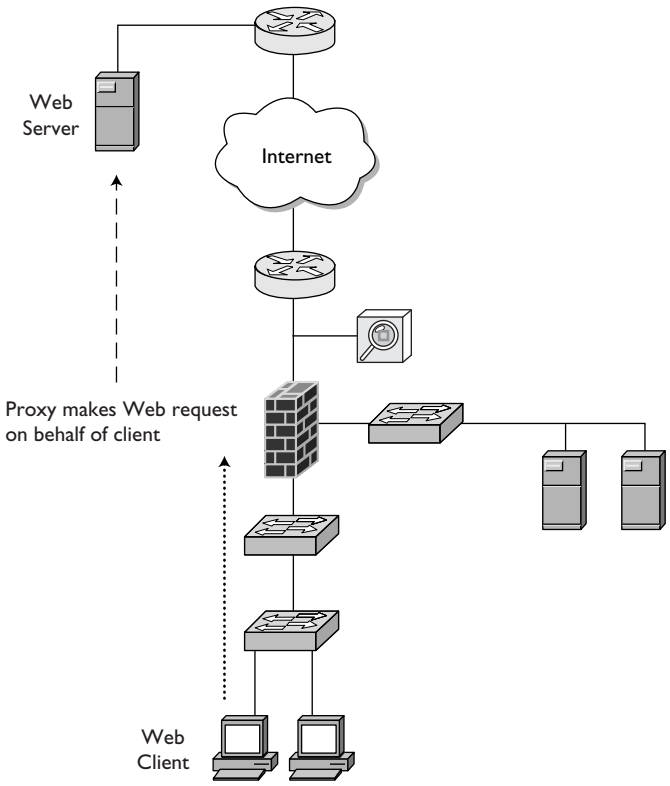


Figure 4–19 Forward Proxy Carries Web Request

from this scenario? Without using a reverse proxy, intruders are free to send attacks through SSL, while the IDS sits blindly by. With a reverse proxy deployed, the IDS can monitor transactions between the reverse proxy and the Web server. Traffic sent over the Internet between the Web client and reverse proxy are protected from prying eyes by SSL. In my opinion, this design implements the best of both worlds.

Setting up the reverse proxy is not difficult. In Chapter 2 we built a forward Squid proxy on NetBSD. Here, we use FreeBSD 5.4 RELEASE and an updated ports tree. First, we install Squid and ensure it is built with support for SSL.

```
# cd /usr/ports/www/squid
# make WITH_SQUID_SSL=YES && make install
```

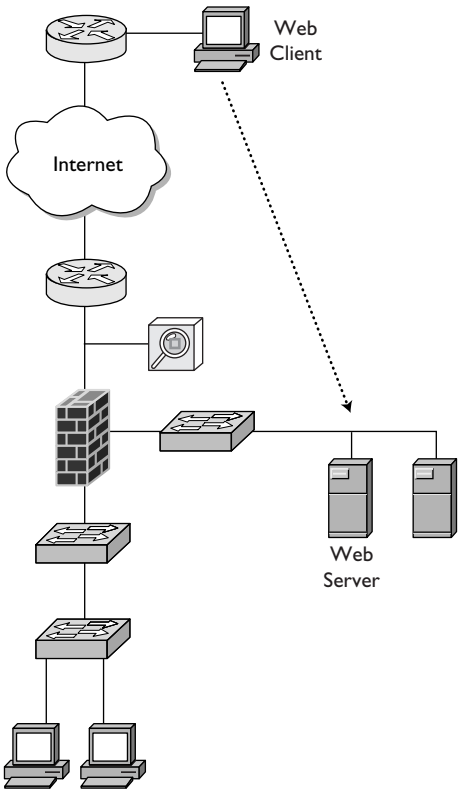


Figure 4–20 Common Web Server Deployment

Next, we configure the /usr/local/etc/squid.conf file by ensuring these additions are made.

```
http_port 80
https_port 443 cert=/usr/src/crypto/openssl/demos/tunala/
 A-server.pem
acl all src 0.0.0.0/0.0.0.0
http_access allow all
httpd_accel_host 192.168.2.15
httpd_accel_port 80
httpd_accel_single_host on
httpd_accel_uses_host_header off
```

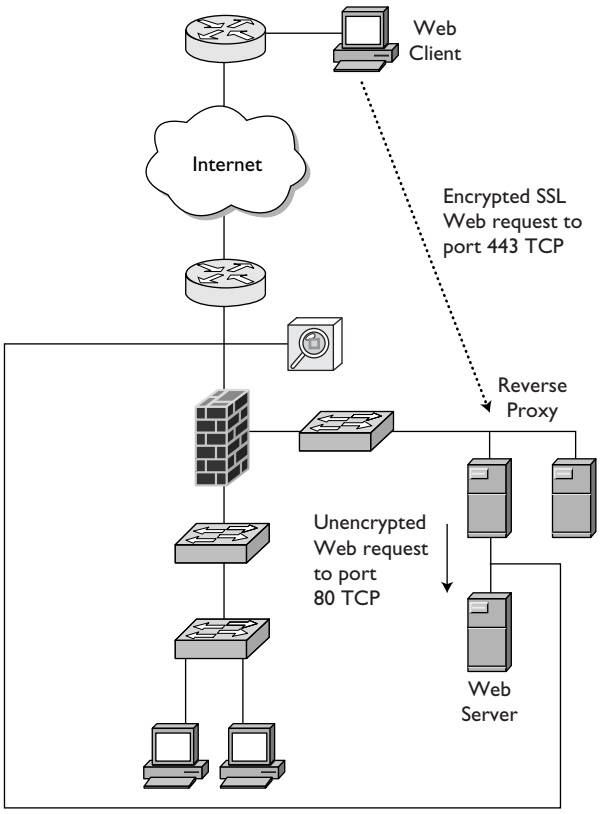


Figure 4–21 Reverse Proxy Terminates SSL Connection

The following explains each line:

1. Line one tells Squid to listen on port 80 TCP. The default is 3128 TCP. Since Web clients do not know they are talking to a reverse proxy, they expect to reach Web servers on port 80 TCP.
2. Line two tells Squid to also listen on port 443 TCP to accept SSL connections. In this demonstration file, we are using a demo SSL certificate installed on FreeBSD 5.4. In reality, you would deploy your Squid reverse proxy with the SSL certificate you purchased for your Web server.
3. Line three defines the all variable to represent any IP address.

4. Line four lets any host on the Internet connect to our proxy. Remember we are setting up the reverse proxy to accept inbound HTTPS connections that are then relayed to the real Web server.
5. Line five tells Squid where to send inbound connections. In this case the Web server has a private internal IP address, 192.168.2.15.
6. Line six tells Squid to send traffic to port 80 TCP.
7. Line seven tells Squid it is proxying for a single back-end Web server.
8. Line eight tells Squid we are not using domain-based virtual hosts based on HTTP/1.1.[5]

It is important to realize that the public IP address assigned to the Squid reverse proxy should be the IP address previously used by the Web server. In other words, a Web client on the Internet requesting www.taosecurity.com should receive the IP address of the Squid reverse proxy. When the proxy receives a connection from the Web client, Squid will send the Web request to the real Web server at 192.168.2.15. All of this is completely transparent to the Web client.

As we did with the forward proxy, we must first start Squid with the `squid -z` command to create caches. Once Squid is running, it will accept connections on ports 80 or 443 TCP and relay them in clear text to port 80 TCP on 192.168.2.15.

CONCLUSION

This chapter demonstrated a variety of novel ways to gain access to network traffic. In the next chapter, we look at methods of employing intranet routing to direct suspicious traffic toward monitoring platforms.

5. For more information on using and configuring Squid, see Duane Wessels, *Squid: The Definitive Guide* (Cambridge, MA: O'Reilly, 2004).