Global Information Assurance Certification Paper

IDS Performance in a Complex Modern Network: Hybrid Clouds, Segmented Workloads, and Virtualized Networks

GIAC Certified Intrusion Analyst (GCIA)

Principal Investigator: Brandon Peterson
Advisor: Dr. Johannes Ullrich
Date: 6/13/2017

Abstract

Most modern networks are complex with workloads in both the cloud and on the premise. Monitoring these types of networks requires aggregating monitoring data from multiple, diverse locations. The following experiment tests the effects on a Snort IDS sensor when monitoring data is sent to the Snort sensor using three different methods. The first method tests direct communication from a server generating test traffic to an IP address on the Snort sensor. The second method captures test traffic from a SPAN port and directs it to an interface on the Snort sensor. The final method simulates ERSPAN by creating a GRE tunnel between the generating server and the Snort sensor and capturing traffic from that tunnel. The results showed that these methods of sending data have a significant impact on the volume of data that reaches the sensor. Also, monitoring can have cascading effects on the network and must be planned for accordingly. For example, when both ERSPAN and production traffic are sent over the same network infrastructure, excessive ERSPAN traffic can cause production traffic to be dropped by overloaded network equipment. When setting up IDS sensors in a complex network environment using SPAN or ERSPAN, it is best to slowly increase the volume of monitoring traffic and carefully measure the impact in each unique environment.

1. Introduction

In recent years, data centers have become increasingly reliant on public cloud infrastructure. Cisco predicts that, "By 2020, 92 percent of workloads will be processed by cloud data centers; 8 percent will be processed by traditional data centers" (Cisco, 2016). Monitoring cloud based networks with existing IDS/IPS infrastructure requires feeding the IDS sensors using techniques such as ERSPAN to encapsulate and send the monitoring traffic between disparate networks.

Micro-segmented networks increase security by compartmentalizing computers into zones or enclaves based on their function. For example, in a micro-segmented network, human resource machines are separated from HVAC systems. As traffic traverses across the boundaries between segments, it is far easier to capture and inspect that traffic with an IDS than in flat networks. SPAN and RSPAN offer cost effective options for organizations to capture and inspect this traffic.

Snort is a good option to test the impact that various network configurations have on IDS performance due to its wide acceptance in the marketplace. According to a 2014 Technology Brief by Cisco: "With nearly 4 million downloads and hundreds of thousands of registered users, Snort is the most widely deployed IPS technology in the world" (Cisco, 2015). Snort has proven itself an effective IDS/IPS choice with its signature, protocol, and anomaly-based protection. Snort also offers the ability to write custom signatures. Unfortunately, Snort does have its drawbacks; the chief among these is the fact that its inspection engine is single-threaded. Single-threaded processes only utilize a single CPU and are not able to spread their workload across multiple CPUs or cores. The Snort process will drop packets or pass traffic through uninspected if the CPU usage gets too high. To overcome this limitation, multiple instances of Snort are typically run in parallel and the traffic is split between those processes using a load balancer such as pf_ring.

For Snort to function as an IDS, it needs to receive a copy of the traffic to inspect. There are several methods organizations can choose from to accomplish this. Each method has its pros and cons. Network taps, SPAN, RPSAN, and ERSPAN represent some of the common methods used to copy traffic to an IDS sensor for inspection.

1.1. Tap

The ideal method of feeding an IDS sensor is with a network tap. According to Gigamon, "A network tap is a hardware component that connects into the cabling infrastructure to copy packets" (Gigamon, 2017). In the case of a copper cable, the tap is powered and makes two copies of the source packets. One copy is sent to the destination and one is sent to the IDS sensor. A fiber tap is simpler—it contains a mirror that splits the light between two destination ports. Using a tap allows sensors to be placed throughout the network without impacting the performance of production switches or servers. Taps are expensive and installing them at every point on the network an organization chooses to monitor can quickly become cost prohibitive. Also, in cloud environments, installing a tap may not be an option, thus, additional methods of feeding IDS sensors are necessary.

1.2. SPAN

SPAN ports serve the same function as a tap, however, they are created using existing switches rather than using a dedicated piece of hardware. Gigamon's literature explains that "A SPAN (Switch Port Analyzer) is a software function of a switch or router that duplicates traffic from incoming or outgoing ports and forwards the copied traffic to a special SPAN (or sometimes called mirror) port." (Gigamon, 2017). The simplicity and effectiveness of SPAN ports allow analysts to deploy IDS sensors throughout the network with minimal infrastructure cost. Since SPAN ports are a software function of a switch, they may have "a negative impact on switch performance (despite what the manufacturer may claim). Depending on the specific features of its design and on network traffic, switch ports can slow down their operation." (Lukatsky, 2003). For example, when the switch is overloaded, traffic on the SPAN port is typically dropped first (Delaney, 2013). An organization using SPAN ports must thoroughly test both the performance of the IDS sensor and the impact of the SPAN port on the switch. Otherwise, a sophisticated attacker might be able to overload the switch and bypass IDS.

1.3. RSPAN

A SPAN port requires the sensor to be plugged into a port on the same switch providing the SPAN. RSPAN (Remote Switch Port Analyzer) overcomes this limitation. Using RSPAN, source traffic from one switch can be sent across the local area network to a destination port on a different switch. This allows organizations to place their IDS sensor in a convenient location on the LAN rather than directly connected to the source switch.

RSPAN shares some of the same limitations of SPAN; it relies on the switch or router to copy the packets to another port. This can lead to overloaded equipment and dropped packets. In addition, RSPAN strips off the original VLAN ID of the packet and replaces it with the RSPAN VLAN ID. In a network that re-uses IP addresses (separated by VLAN tagging), determining the actual source of the traffic will be more difficult.

1.4. ERSPAN

ERSPAN (Encapsulated Remote Switch Port Analyzer) typically uses GRE (Generic Routing Encapsulation) to package the mirrored traffic for transport across wide area networks. Effectively, mirrored traffic can be sent from nearly any network to the organization's sensors. Also, many IDS sensors and network security tools can de-encapsulate GRE traffic natively. For example, both Wireshark and Tenable's Passive Vulnerability Scanner have this capability. Having tools that can decode GRE packets natively simplifies configuration by eliminating the need for a destination switch to decode the traffic.

The GRE tunnel will most likely be running across the normal production network, potentially impacting network performance. Without some extra layer of QoS (Quality of Service) during periods of high traffic, the ERSPAN traffic could cause normal traffic on overloaded network equipment to be dropped or slowed. Also, there are extra computations required to encapsulate and de-encapsulate the mirrored traffic.

2. Test Environment and Methods

Three different methods of delivering data for inspection to a Snort sensor were tested: direct connection, SPAN, and GRE tunnel to simulate ERSPAN. Dedicated hardware and the latest stable release of all software was used in the lab.

2.1 Hardware and Software Setup

CentOS 7.1 was installed on a Dell PowerEdge R430 to function as the traffic generator. The server, named tcpreplay, was connected to a Brocade ICX 7750 at 10Gb. Citrix XenServer 7.1 was installed on a Dell PowerEdge R710 and two virtual machines running CentOS 7.1 were created to act as the Snort sensor and a Dummy server. The Citrix XenServer was connected to the Brocade ICX 7750 at 1Gb. Each virtual machine was given 4 vCPUs and 8GB RAM. The hardware configuration for the test environment is listed below in Table 1:

Tcpreplay server	Snort Sensor (Virtual Machine)	Dummy Server (Virtual Machine)	Switch
Dell PowerEdge R430 2 x Intel Xeon CPU E5-2630 @ 2.2GHz 64Gb Memory Broadcom BCM57810 10 Gb Ethernet	Dell PowerEdge R710 4 vCPU out of 2 x Intel Xeon CPU E5645 @ 2.40GHz 8GB out of 48GB Memory Broadcom BCM5709 1Gb Ethernet	Dell PowerEdge R710 4 vCPU out of 2 x Intel Xeon CPU E5645 @ 2.40GHz 8GB out of 48GB Memory Broadcom BCM5709 1Gb Ethernet	Brocade ICX 7750 48-port 10GB 6-port 40Gb

Table 1: Hardware Configuration

On the Tcpreplay server, tcpreplay 4.2.1 was installed. On the Snort sensor, Snort 2.9.9.0 was compiled from source and installed. Snort was compiled with the "--enable-non-ether-decoders" option so it could process packets from the GRE tunnel interface used in the ERSPAN simulation. Table 2 lists the key software components installed for the test environment:

Tcpreplay server	Snort Sensor	Dummy Server	Switch
CentOS 7.3.1611 tcpreplay 4.2.1	CentOS 7.3.1611 VM Snort 2.9.9.0	CentOS 7.3.1611 VM	Version 08.0.30gT201 Switch Code

Table 2: Software Configuration

CentOS 7.1 was installed on the Dummy server. The function of the Dummy server was to serve as a target for traffic that the Brocade switch mirrored to the Snort Sensor in the SPAN simulation. The Dummy server had a firewall rule to silently drop all traffic to minimize the load on the host Citrix XenServer.

Iperf3 was run between the Tcpreplay server, Dummy server, and the Snort sensor to confirm a reliable 1Gb connection. All iperf3 tests approached the 1Gb, averaging 944Mbps.

2.1 Test Traffic

Nmap and tcpdump were used to create a packet capture of a simple TCP SYN scan. The reason this scan was chosen is best described by the tool's author:

"SYN scan is the default …. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections. SYN scan works against any compliant TCP stack rather than depending on idiosyncrasies of specific platforms as Nmap's FIN/NULL/Xmas, Maimon and idle scans do" (Lyon, 2011)

The Snort sensor was verified to be functioning properly by sending a single pass of the pcap file through the Snort sensor and checking that Snort processed, detected, and alerted as expected. The verification process was completed for each method used.

2.3 Direct Connection Method

The Snort server was given an IP address on a dedicated interface for the Direct Connection method. The Tcpreplay server was also given an IP address using a dedicated NIC on the same network. Figure 1 shows a direct connection between the host Tcpreplay generating the traffic and the Snort sensor:

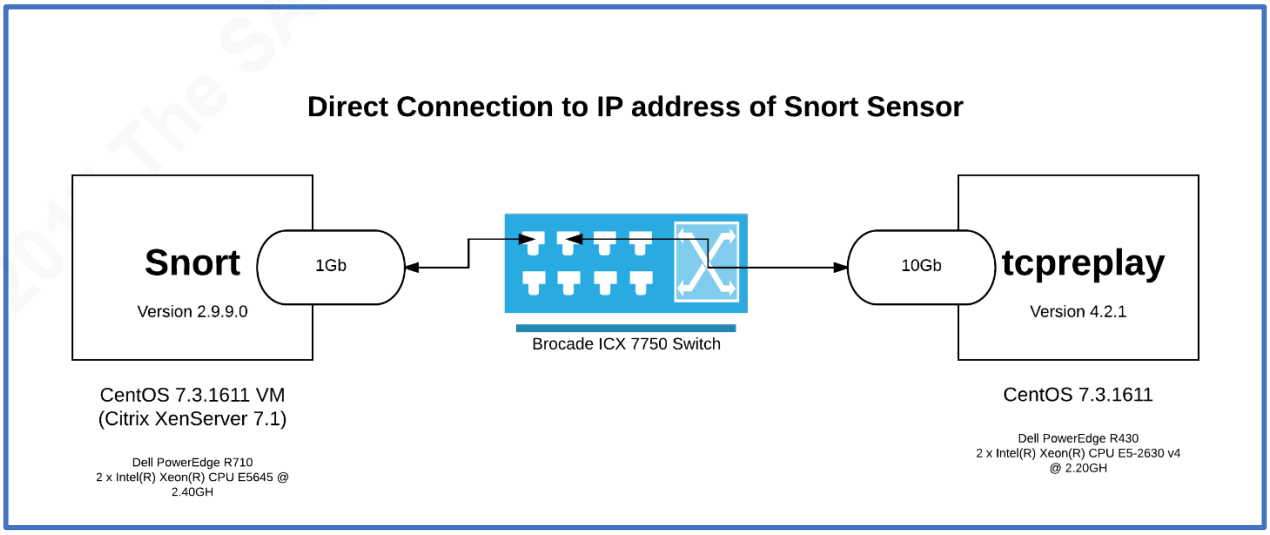

Figure 1:Direct Connection to Snort

Snort was started with the "-p" option to prevent the adapter from going into promiscuous mode. This prevents the sensor from processing any packets that were not directly sent to the sensor. The purpose of the Direct Connection method is to simulate a network tap as close as

possible. The Direct Connection method will be used as the baseline by which the other methods will be compared.

2.4 SPAN Method

Traffic to and from the Dummy server was duplicated by the Brocade switch and was sent to an interface on the Snort sensor using the SPAN method, seen in Figure 2:

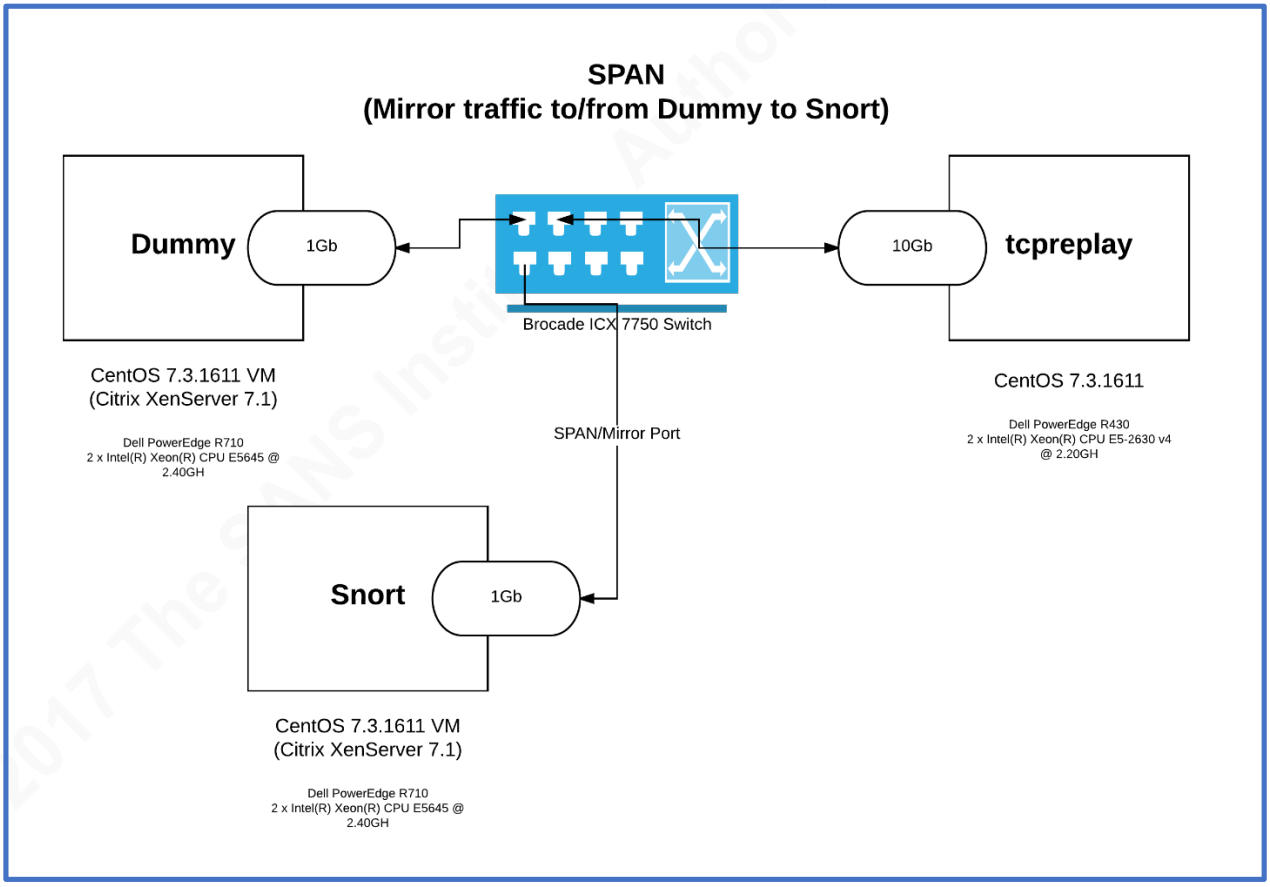

Figure 2: SPAN Connection Method

The Snort sensor received a copy of the pcap traffic the Tcpreplay server sent to the Dummy server as well as the Dummy server's response. A firewall rule was created to drop all traffic on the Dummy server to keep processing overhead on the XenServer host to a minimum.

2.5 ERSPAN (GRE Tunnel) Method

A GRE tunnel was setup between the Tcpreplay server and the Snort sensor to simulate an ERSPAN connection. Figure 3 shows how the pcap traffic was then sent directly to the tunnel interface:

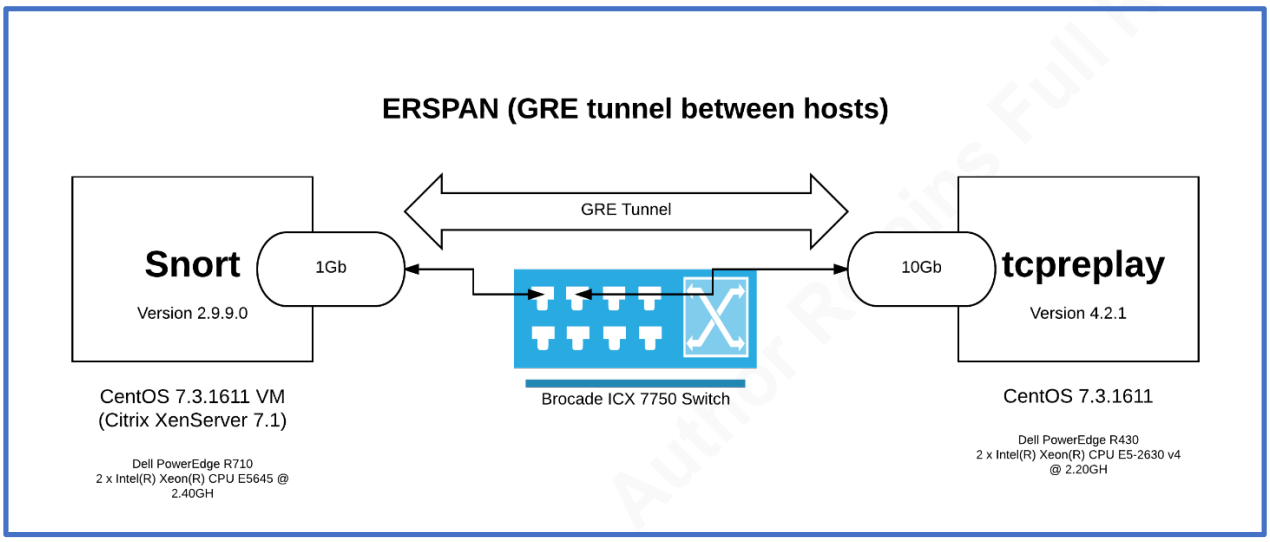

Figure 3: ERSPAN Connection Method

In this setup, the switch is simply used to transport the packets between the Tcpreplay and Snort servers. In the test environment, the switch does not perform any of the encapsulation/de-encapsulation for the GRE tunnel. Most enterprise-grade switches can, however, be used as either tunnel endpoint for ERSPAN.

2.6 Measurements

The pcap file was replayed for one hour at three speeds: 10Mbps, 100Mbps, and 400Mbps to measure the performance of Snort. For each speed, maximum memory usage percentage, maximum CPU usage percentage, the total number of packets processed, and the total number of packets dropped were captured. Snort will begin dropping packets when CPU usage is too high. Snort cannot handle as many tcp and udp connections or run as many protections if available memory is too low.

3. Results

For each of the test methods, memory usage, CPU usage, packets dropped, and total packets processed taken from the Snort sensor was recorded. Except for the memory usage, the measured results were unexpected and inconsistent with initial expectations that each of the methods used to feed Snort would result in nearly identical measurements at any given tcpreplay throughput. Testing showed that the tcpreplay "Rated" speeds did not accurately represent the

volume of traffic reaching the Snort sensor. Due to the unexpected results, each round of experiments was run multiple times to ensure accurate measurements.

3.1 Memory Usage

Snort was very consistent in the memory usage tests. As the tests ran, Snort would slowly consume more and more memory. Figure 4 shows the percentage of system memory the Snort process is using as the tcpreplay playback speed is increased for the Direct, SPAN, and ERSPAN communication methods:

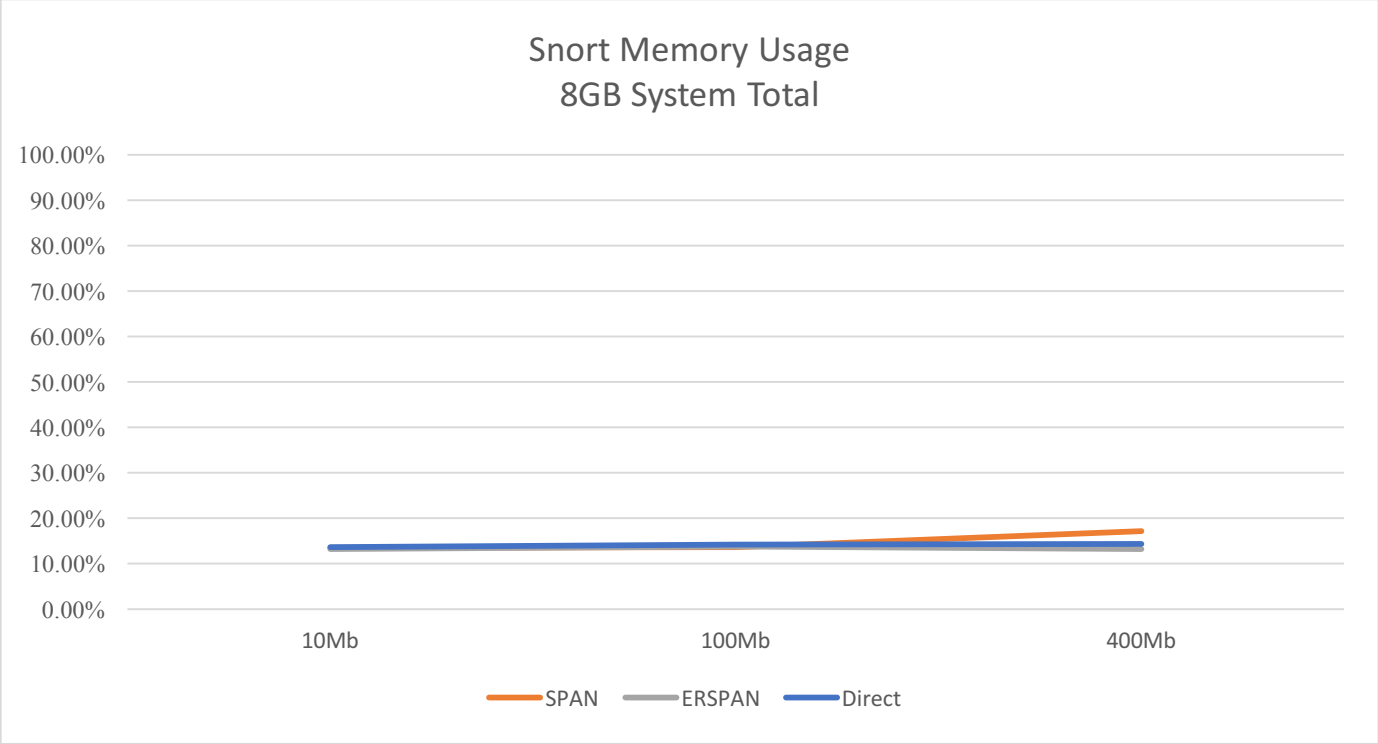

Figure 4: Snort Memory Usage

Even with tcpreplay running at 400Mbps for over an hour, Snort never went above 20% memory usage—even as Snort was CPU bound. The results show that the method of communication does not have a significant impact on the memory usage of Snort. This means that organizations do not need to devote time to testing Snort memory usage under most circumstances. A sufficiently equipped machine should not run into memory issues running Snort.

3.2 CPU Usage

CPU usage proved to have the most impact on Snort performance. As Figure 5 shows, CPU usage was inconsistent between methods at each tcpreplay speed:

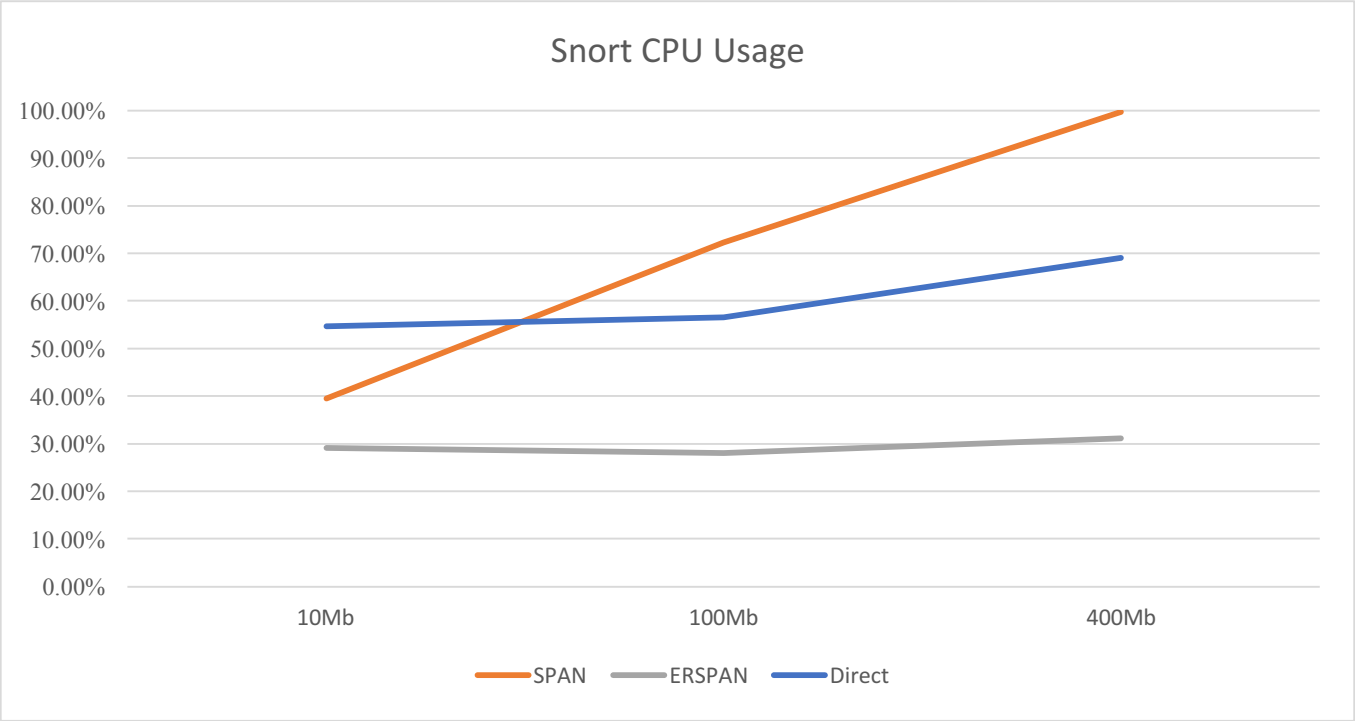

Figure 5: Snort CPU Usage

The Direct Connection method held steady at below 60% CPU utilization and only reached 70% near tcpreplay's maximum transmission rate. The SPAN rate climbed in near linear fashion starting at 40% CPU usage (10Mbps) and continuing to 100% usage (400Mbps). Finally, the ERSPAN rate remained nearly unchanged across all replay speeds, at around 30%.

Looking at the percentage of packets dropped during each one hour run shows that only the SPAN method dropped significant packets as displayed in Figure 6:

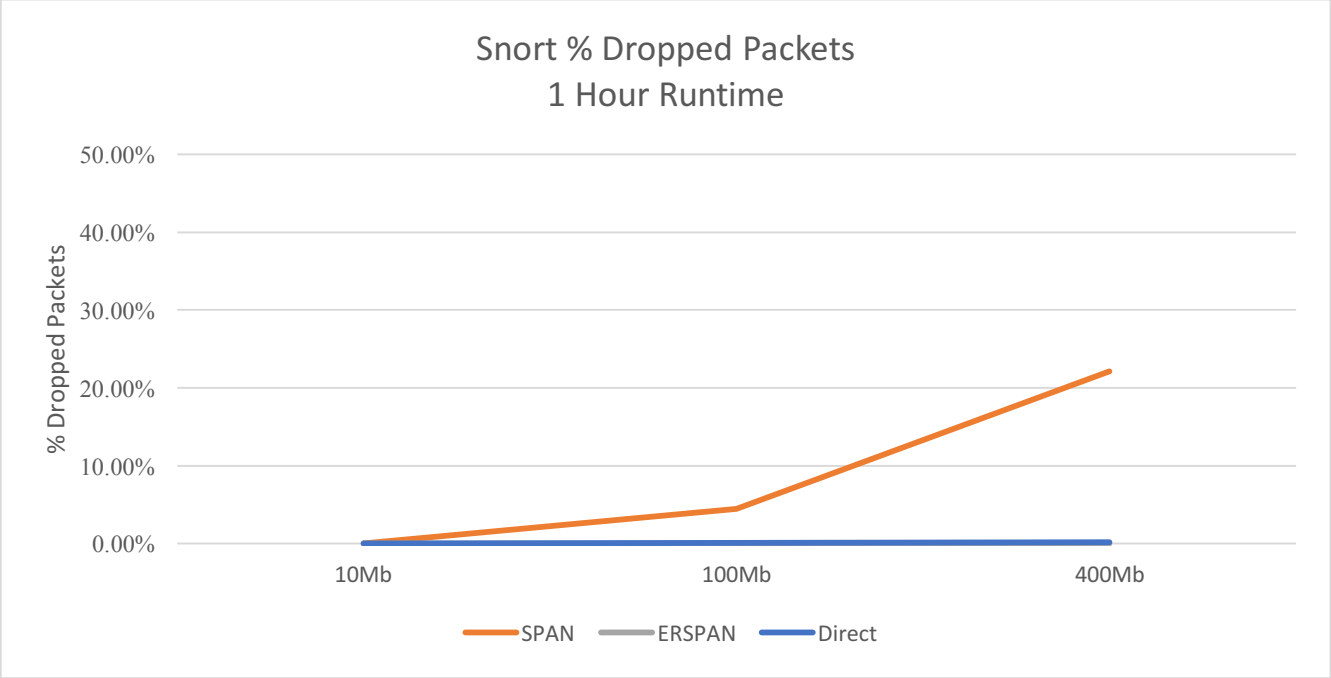

Figure 6: Snort Packet Loss

Figure 7 shows the correlation between the percentage of packets dropped and the percentage CPU usage of the Snort process:

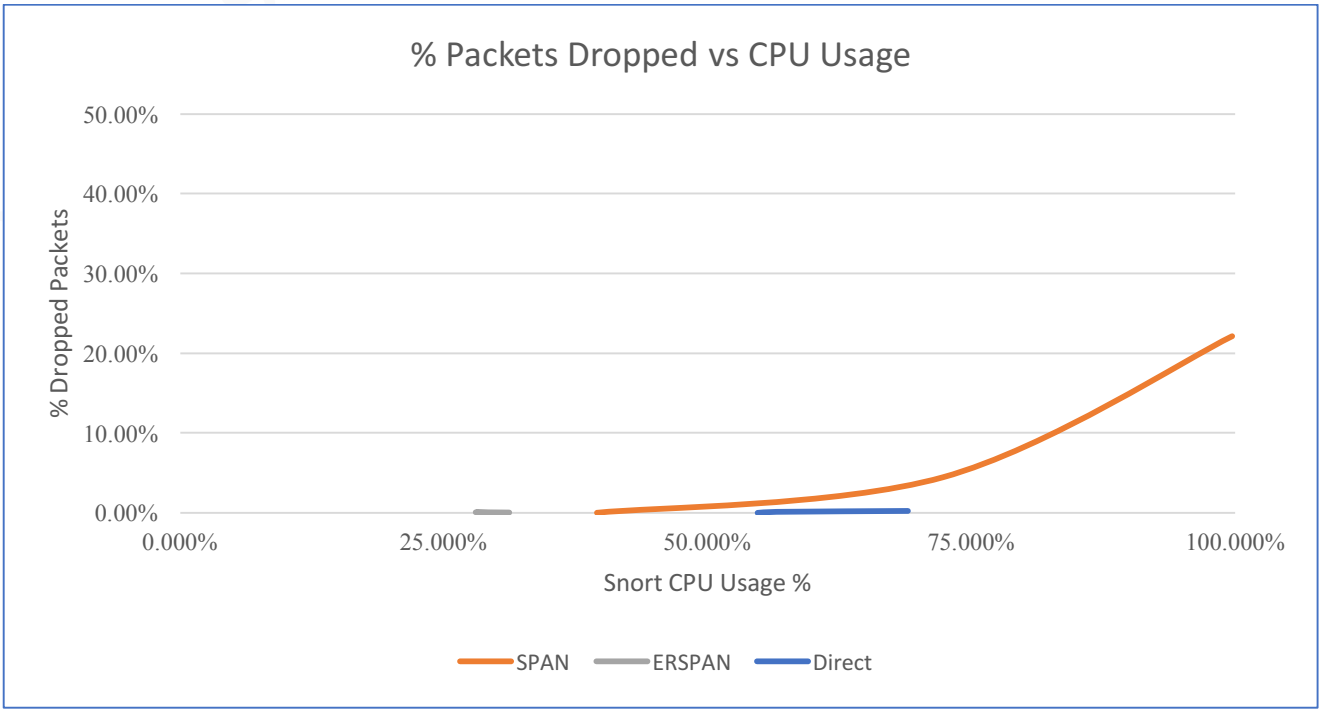

Figure 7: Snort Packet Loss vs CPU Usage

With the SPAN method, as the CPU Usage climbed, so did the number of dropped packets. At 400Mbps the Snort sensor was at 100% CPU utilization and dropping over 22% of the packets it observed. As a single-threaded, processor-intensive application, this is the expected behavior. The Direct and ERSPAN method did not drop significantly more packets even as the throughput increased to 400Mbps.

As seen in Figure 8, there are large variations in total packets that Snort processed, even though tcpreplay reported that it had sent packets at the prescribed rates of 10Mbps, 100Mbps, and 400Mbps:

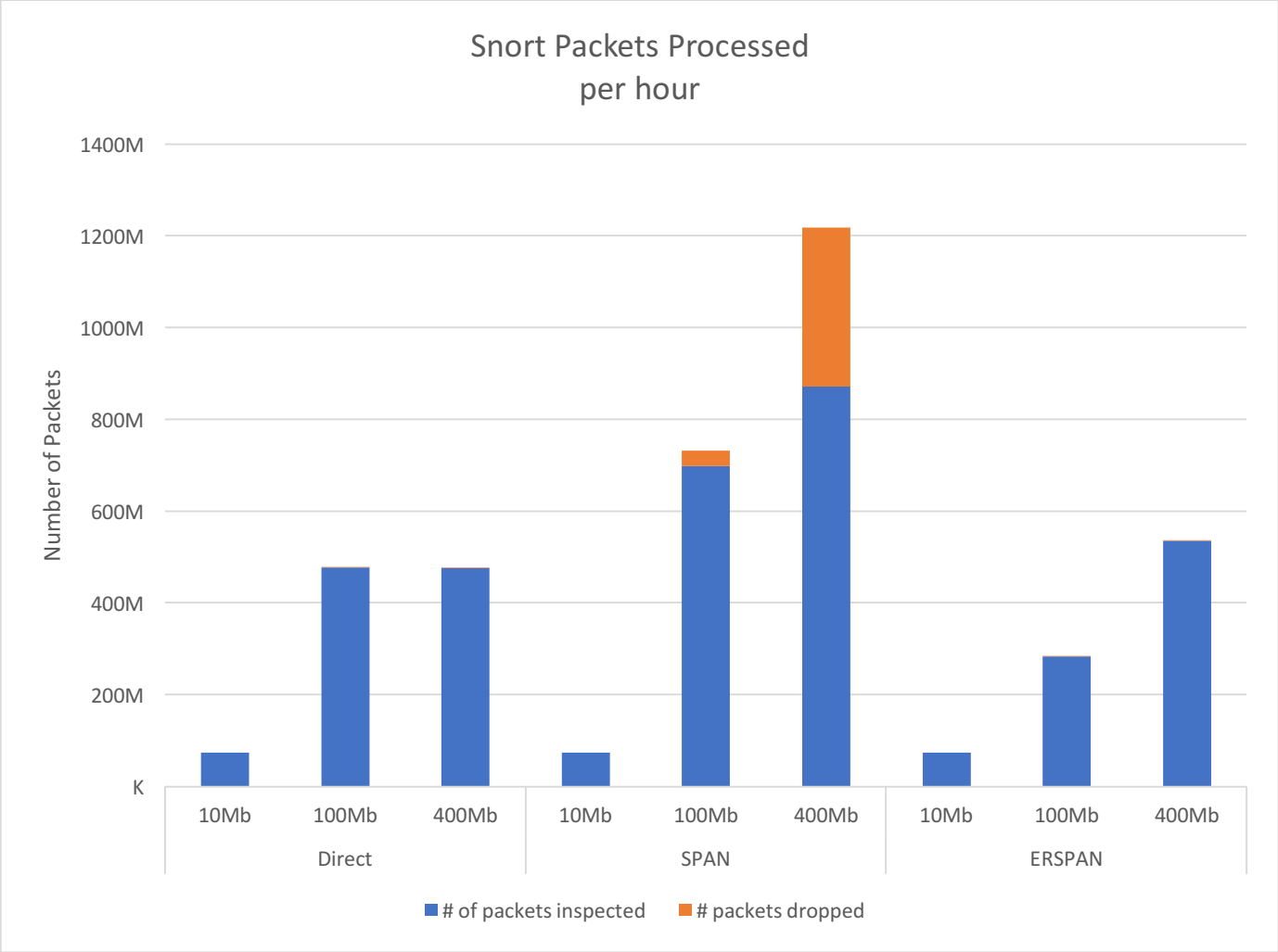

Figure 8: Snort Packets Processed

The results showed that the SPAN method processed significantly more packets than the other two methods. This explains why the SPAN method eventually dropped significantly more packets and ran the CPU to 100% utilization.

4. Conclusion

It is clear from these results that the Snort process itself is not directly impacted by the method of feeding the sensor, whether from a direct connection, SPAN, or ERSPAN. However, the rate at which packets can be fed to the sensor fluctuates dramatically depending upon the method used.

The direct connection had the lowest capacity of the methods tested. Tcpreplay, using the direct connection, was only able to send about 475 million packets per hour at 400Mbps. Using ERSPAN, tcpreplay could transmit around 535 million packets per hour at 400Mbps. Even with the extra overhead of de-encapsulating the GRE tunnel, ERSPAN processed 12.6% more packets than the direct connection method. The SPAN method could process nearly 700 million packets before suffering significant packet loss. This is 30.8% more packets processed than the ERSPAN method and 47.4% more than the direct connection method.

In testing, the Snort process safely handled over 100Mbps of traffic under each method. Therefore, a single modern multi-core x86 server will be able to run enough Snort processes to easily inspect 1Gb/sec of traffic or more. However, the ERSPAN and direct connection are limited in their throughput. They both show far fewer packets processed at the 100Mbps transfer speed. Both ERSPAN and the direct connection method would only be reliable at the slowest speeds: 10Mbps or less. Unfortunately, this limits their usefulness in most modern networks operating at higher speeds. Using SPAN, the switch never showed a significant CPU load or dropped any packets while mirroring traffic at the maximum tcpreplay speed of just over 400Mbps.

While hardware taps are the preferred option, the test results have validated SPAN as a viable alternative for network monitoring. The other methods tested, direct connection and ERSPAN, have limited usefulness and should only be considered in low throughput, remote environments.

Brandon Peterson, brandonp@dri.edu

5. Limitations

Access to a network tap was not available at the time of testing. Instead, packets were sent directly to the addressed interface on the Snort sensor, with Snort using the –p option. This prevents Snort from putting the interface into promiscuous mode, therefore only analyzing packets sent directly to the Snort sensor's IP address. There was no additional traffic on the test network and both machines were connected through a high-end switch (Brocade ICX 7750), which Brocade claims has a switching capacity of 1.92 Tbps (Brocade, 2017). Unfortunately, even with a dedicated high-end switch, the direct connection results were much slower than expected. A dedicated hardware tap would operate at the speed of the source.

Other hardware issues included only having a single high-end switch capable of running RSPAN to test with. Instead of using hardware, RSPAN was configured using the Citrix Distributed Virtual Switch Controller software on a single Citrix XenServer server. Unfortunately, with this configuration, all the traffic for both the RSPAN VLAN and the virtual machines ran on the same virtual server. In testing, the throughput for the RSPAN and normal traffic interfaces was less than 200Mbps each—far lower than the 475Mbps tcpreplay could send directly. At the lower throughput, Snort had no trouble keeping up with the traffic volume. Dropped packets never went above 0.4%, which is in the range of the direct connection. Without the capability to compare RSPAN against the direct connection method, SPAN, and ERSPAN at higher speeds, the results were not included. Even though RSPAN was not tested at higher speeds, it was successfully tested up to 200Mbps. Considering this, RSPAN warrants consideration for organizations monitoring traffic on the edges of their local area network.

The lack of a second high-end switch also meant that the GRE tunnel used for ERSPAN was setup directly between the Tcpreplay server and the Snort sensor using the Linux GRE kernel module, ip_gre. GRE encapsulates each packet with 24 extra bytes of data. For every 1MB sent across a network with an MTU of 1500 bytes, an extra 16KB of encapsulation data is sent. Packet fragmentation can also become an issue if the MTU size is not increased. For example, a 1,500 byte packet would need to be fragmented into two packets to traverse a GRE tunnel if the MTU size of the network was not increased (Hogg, 2013). The extra encapsulation data, packet fragmentation, as well as, the software based encapsulation/decapsulation process used in the test environment, led to lower throughput levels.

Software was also a limiting factor during testing. Tcpreplay, using the –topspeed option, only reached speeds around 475Mbps--even on a high-end server with a 10Gb NIC. The tcpreplay website addresses the speed issue:

> "The following is known to apply to the 2.4.x and 2.6.x series of kernels. By default Linux's tcpreplay performance isn't all that stellar. However, with a simple tweak, relatively decent performance can be had on the right hardware…On one system, we've seen a jump from 23.02 megabits/sec (5560 packets/sec) to 220.30 megabits/sec (53212 packets/sec) which is nearly a 10x increase in performance. Depending on your system and capture file, different numbers may provide different results" (tcpreplay Wiki, 2012).

Tcpreplay speed was capped at 400Mbps to ensure consistency across test runs due to the speed issue.

In testing Snort, additional software related issues arose depending on the operating system chosen to test on. First, using both the SecurityOnion and Ubuntu, SIGUSR1 (Control-C) signal could not be reliably sent to the Snort process. SIGUSR1 is necessary to print that Snort statistics: packet loss, number of packets processed, etc. The same behavior was found whether Snort was installed from source or pre-compiled binaries. Snort was then installed on Windows 7. Snort ran without issue on Windows and properly accepted the Control-C combination to display statistics. However, the Windows shell does not have a method to script sending Control-C to the Snort process. This made the Windows version impractical for testing. Finally, Snort was compiled from source and installed on CentOS 7.1. Sending the Snort process SIGUSR1 on CentOS worked flawlessly. This allowed the test runs to be scripted and run consistently across multiple test runs.

6. Future Work

Tcpreplay reported identical transfer rates between test methods despite the differences reported by Snort in the total number of packets processed (received. Given that the same packet capture file was used, at the same speed and duration, Snort should have had identical packet totals. However, different total packets processed was observed between the Direct connection, SPAN, and ERSPAN tests. Future work would include accounting for these differences. It may

be due to the limitation of the underlying hardware or an issue with tcpreplay. Given iperf3 reported near 1Gb connection speeds and the previously mentioned tcpreplay issues, tcpreplay's reported speed measurements are questionable. Future work would include verifying the tcpreplay rates or finding a reliable alternative to tcpreplay.

Brandon Peterson, brandonp@dri.edu

References

Brocade. (2017). *Brocade ICX 7750 Switch Data Sheet*. Retrieved from http://www.brocade.com/en/backend-content/pdf-page.html?/content/dam/common/documents/content-types/datasheet/brocade-ruckus-icx-7750-ds.pdf

Cisco. (2016). *Cisco Global Cloud Index: Forecast and Methodology, 2015–2020*. Retrieved from http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf

Cisco. (2015). *Snort: The World's Most Widely Deployed IPS Technology*. Retrieved from http://www.cisco.com/c/en/us/products/collateral/security/brief_c17-733286.html

Delaney, Darragh. (August 2013). *The Do's and Do NOT's of using SPAN Ports*. Retrieved from http://www.lovemytool.com/blog/2013/08/the-dos-and-do-nots-of-using-span-ports-by-darragh-delaney.html

Gigamon. (2017). *Whitepaper: Understanding Network TAPs – The First Step to Visibility*. Retrieved from https://www.gigamon.com/sites/default/files/resources/whitepaper/wp-understanding-network-taps-the-first-step-to-visibility-3164.pdf

Hogg, Scott. (2013). MTU Size Issues, *Network World*. Retrieved from http://www.networkworld.com/article/2224654/cisco-subnet/mtu-size-issues.html

Lukatsky, A. (2003). *Protect your information with intrusion detection*. Wayne, PA: A-List.

Lyon, Gordon. (2011). The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure.Com LLC. https://nmap.org/book/toc.html

Tcprelay Wiki. (2012). Retrieved from http://tcpreplay.synfin.net/wiki/tcpreplay