Possible Packet Loss during reassembly for Snort IDS/IPS sensors:

Originally Written on November 29, 2014.

Last Revised on February 12, 2015.

This document describes possible issues with the SNORT IDS and packet reassembly issues when SNORT is operating in IPS (inline) mode.

Blue – Informational Messages and Comments
Orange – These are commands the user types at the shell prompt
Red – Read Carefully before proceeding.

Recently, some users on the SNORT mailing list posted questions on packet loss while snort (**www.snort.org**) was operating in IPS (inline) mode, and in doing some research, this is what I have found (some of this information is from the wireshark wiki) and other information was added to the latest release of the snort user's manual for Snort 2.9.7.0 in the doc/INSTALL file.

From the Snort INSTALL Manual:

Potential Problems between a Network Interface Card (NIC) and Snort
==========

There are two common problems that may occur when installing/running Snort for the first time.

The first is that the NIC is not in **promiscuous** mode (which allows the network card to 'see' all of the network traffic sent to it, rather than just traffic which is destined for the IP address and/or network mask assigned to the NIC. That means traffic may be stopped at the NIC and in such cases, Snort **will not** see the packet data.

To resolve this problem, **enable promiscuous mode** on the interface which Snort is running. On Linux and Unix based systems, this may be done via the '**ifconfig**' command as the 'root' or super user.

A second common problem is that NIC's found in PC's and Servers have features that may cause problems in Snort's Stream5/6 packet reassembly. For instance, when **Large Receive Offload** (**LRO**), **Large Send Offload** (**LSO**), or **Generic Receive Offload** (**GRO**) is enabled, the NIC will reassemble packets without the aid of the PC or Server CPU. The reassembled packets, which may be **significantly larger** than the **standard snaplen/MTU** of **1518 bytes**, are then sent to the operating system and Snort. However, Snort will truncate packets larger than the default **snaplen/MTU** of **1518 bytes**.

****** WARNING ******

Before making any changes to production hardware, it is strongly recommended that you test the instructions below in a lab environment or non-mission critical hardware.

****** WARNING ******

Also, this is by no means a full list of parameters that may cause issues with Snort. This document is simply a guide suggesting places to start looking when snort appears to drop or block traffic for reasons that is not explained by configuration issues or enabled rules in Snort.

Parameters which may impair packet reassembly for SNORT:

***** OFFLOADING *****

Most modern operating systems support some form of network offloading, where some network processing happens on the NIC instead of the CPU. Normally this is a great thing. It can free up resources on the rest of the system and let it handle more connections. If you're trying to capture network traffic it can result in false errors and strange or even missing traffic.

In SNORT, this may cause false alerts or missed network traffic being detected or stopped by SNORT (if operating via inline (IPS) mode).

***** CHECKSUM OFFLOADING *****

On systems that support checksum offloading, IP, TCP, and UDP checksums are calculated on the NIC just before they're transmitted on the wire.

This may hamper the ability for snort to operate properly in IPS (inline) modes and may manifest in packet processing statistics as showing up as packet loss issues to Snort. Additionally, a modern operating system may not bother to initialize the data, so data might be reported as suspect in SNORT (or other packet capturing tools) when it is actually not (i.e. - possible false positive/negative alert).

If you are experiencing network problems and while trying to figure it out with SNORT and are seeing possible checksum errors, you may have a network card with TCP checksum offload enabled and for some reason the packet is not being fixed by the adapter (NAT, bridge or route redirection is sending the packet to another interface). In this case, you may want to check and disable checksum offload for the adapter, if possible.

***** SEGMENTATION OFFLOADING ******

Some network interface cards can reassemble traffic using features found on the NIC, rather than the CPU of the computer. This may manifest itself in packet capture programs/utilities as packets that are larger than expected, such as a 2900-byte packet on a network with a 1500-byte MTU.

You can check and change offloading behavior on Linux and Windows using methods described in this document.

***** TCP CHIMNEY *****

TCP chimney offloading lets the NIC handle processing for established TCP connections. On Windows based systems, offloaded connections may bypass WinPcap, which means that you won't capture these established TCP conversations.

How to check to see if these features are enabled/disabled on various operating systems:

Linux Based Systems

In Linux based systems, you can get a listing of the capabilities of a given network card by using the following command as 'root' or superuser (eth0 is used in this example):

ethtool -k eth0

Which should produce the following output:

Features for eth0:
rx-checksumming: off <--- (ethtool setting is 'rx')
tx-checksumming: off <--- (ethtool setting is 'tx')
tx-checksum-ipv4: off [fixed]
tx-checksum-ip-generic: off
tx-checksum-ipv6: off [fixed]
tx-checksum-fcoe-crc: off [fixed]
tx-checksum-sctp: off [fixed]
scatter-gather: on
tx-scatter-gather: on
tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: off <--- (ethtool setting is 'tso')
tx-tcp-segmentation: off
tx-tcp-ecn-segmentation: off [fixed]
tx-tcp6-segmentation: off [fixed]
udp-fragmentation-offload: off [fixed]
generic-segmentation-offload: off <--- (ethtool setting is 'gso')
generic-receive-offload: off <--- (ethtool setting is 'gro')

large-receive-offload: off [fixed] <--- (ethtool setting is 'lro')
rx-vlan-offload: on
tx-vlan-offload: on [fixed]
ntuple-filters: off [fixed]
receive-hashing: off [fixed]
highdma: off [fixed]
rx-vlan-filter: on [fixed]
vlan-challenged: off [fixed]
tx-lockless: off [fixed]
netns-local: off [fixed]
tx-gso-robust: off [fixed]
tx-fcoe-segmentation: off [fixed]
tx-gre-segmentation: off [fixed]
tx-udp_tnl-segmentation: off [fixed]
tx-mpls-segmentation: off [fixed]
fcoe-mtu: off [fixed]
tx-nocache-copy: on
loopback: off [fixed]
rx-fcs: off
rx-all: off
tx-vlan-stag-hw-insert: off [fixed]
rx-vlan-stag-hw-parse: off [fixed]
rx-vlan-stag-filter: off [fixed]

In addition, if a given parameter has the term [fixed] next to it, that means that the parameter may NOT be adjusted by ethtool.

Frequently, an option is [fixed] because the snort is running is in a virtual environment (i.e. - VMWare, VirtualBox, Xen, etc) and the guest operating is unable to change the host operating system options. Also, to disable these options under Linux, you can use the '**ethtool**' command with the following parameters:

ethtool -K eth0 tx off rx off tso off gso off gro off

If you need more help with ethtool options, you may use '**info ethtool**' or '**man ethtool**'. You may also wish to keep these options disabled across system shutdowns or restarts.

In OpenSuSE 13.x, permanently setting these options can be done as follows (This usually requires **root user** access):

cd /etc/sysconfig/network
cat ifcfg-eth0 (which should produce the listing below):

BOOTPROTO='static'
BROADCAST=''
ETHTOOL_OPTIONS='' // <-- <-- <-- <-- <-- (permanent ethtool options)
IFPLUGD_PRIORITY='0'
IPADDR='192.168.1.40/24'
MTU=''
NAME='82540EM Gigabit Ethernet Controller'
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
USERCONTROL='no'

Next, open up the file ifcfg-eth0 with your favorite text editor and edit the line beginning with **ETHTOOL_OPTION=’’** to look like the following:

ETHTOOL_OPTIONS='-K eth0 tx off rx off tso off gso off gro off'

Now, the options will be permanently turned off. In Fedora 19, the network scripts are located in '**/etc/sysconfig/network-scripts**'.

Other Linux distributions may have similar configurations, or in the case where you are using a distribution which supports '**rc.local**' you can insert the '**ethtool -K**' command into that file as well.

Checksum offloading can be enabled and disabled with the ‘**ethtool**’ command.

To check:

ethtool --show-offload ethX (where X is the interface number, 0, 1, 2, etc)

To disable:

ethtool --offload ethX rx off tx off

Or, with some 3Com cards (see 3c59x vortex docs):

rmmod 3c59x; **modprobe 3c59x hw_checksums=0**

Unix Based Systems

In BSD based systems, the parameters may be changed via the '**ifconfig**' command. For example, in FreeBSD 9.x, to disable transmit checksum offloading (TXO), receive checksum offloading (RXO), and large receive offloading (LRO) for device '**em0**' (ethernet 0) type the following command:

/sbin/ifconfig em0 -txcsum -rxcsum -lro

Enabled options for a given interface appear on the OPTIONS line, as shown below:

/sbin/ifconfig

```
em0:
flags=28943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST,PPROMIS
C> metric 0 mtu 1500
        options=98<VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
        ether 08:00:27:8e:a0:85
        inet 192.168.1.125 netmask 0xffffff00 broadcast 192.168.1.255
        inet6 fe80::a00:27ff:fe8e:a085%em0 prefixlen 64 scopeid 0x1
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
em1:
flags=28943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST,PPROMIS
C> metric 0 mtu 1500
        options=98<VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
        ether 08:00:27:4c:36:49
        inet 192.168.56.125 netmask 0xffffff00 broadcast 192.168.56.255
        inet6 fe80::a00:27ff:fe4c:3649%em1 prefixlen 64 scopeid 0x3
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
        options=600003<RXCSUM,TXCSUM,RXCSUM_IPV6,TXCSUM_IPV6>
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x4
        inet 127.0.0.1 netmask 0xff000000
        nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
```

On interfaces '**em0**' and '**em1**' you will see that the network cards are in PROMISC mode, but **transmit checksum offloading, receive checksum offloading, and large receive offloading are not listed, indicating they are disabled**.

You can add the parameters to the file '**/etc/rc.conf**' so that they will be enabled across system restarts network interface restarts.

Here is a sample **rc.conf** file from FreeBSD 9.1:

```
hostname="barnyard.foobar.com"
ifconfig_em0=" inet 192.168.1.125 netmask 255.255.255.0"
ifconfig em0 promisc -rxcsum -txcsum -lro
ifconfig_em1=" inet 192.168.56.125 netmask 255.255.255.0"
ifconfig em1 promisc -rxcsum -txcsum -lro
defaultrouter="192.168.1.1"
sshd_enable="YES"
# Set dumpdev to "AUTO" to enable crash dumps, "NO" to disable
dumpdev="NO"
```

In the above case, interfaces 'em0' and 'em1' will be set to **promiscuous mode**, and have **transmit checksum offload**, **receive checksum offload**, and **large receive offload** disabled.

In OpenBSD based systems, you can use the command:

```
# /sbin/ifconfig <interface> hwfeatures
```

to get a list of hardware features that your network cards and device drivers support.

On NetBSD based systems, available hardware options for network cards may be listed with the command below:

```
sbin/ifconfig <interface_name>
```

For example:

```
# /sbin/ifconfig wm0

wm0:
flags=8b43<UP,BROADCAST,RUNNING,PROMISC,ALLMULTI,SIMPLEX,MULTICAST
> mtu 1500

capabilities=2bf80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,
      UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Tx,UDP6CSUM_Tx>
      enabled=0
      address: 08:00:27:cf:55:48
      media: Ethernet autoselect (1000baseT full-duplex)
      status: active
      inet 192.168.1.50 netmask 0xffffff00 broadcast 192.168.1.255
      inet6 fe80::a00:27ff:fecf:5548%wm0 prefixlen 64 scopeid 0x1
```

Shows that this interface supports IPv4 checksum offloading (transmit and receive), TCPv4 checksum offloading (transmit and receive), UDPv4 checksum offloading (transmit and receive), and UDPv6/TCPv6 checksum offloading (transmit only).

To disable these features, use the line below:

/sbin/ifconfig wm0 -ip4csum -tcp4csum -udp4csum -tcp6csum-tx -udp6csum-tx

For more help, use '**man ifconfig**'.

Windows Based Systems

Common options found in Windows Based network cards and drivers may include (there may be others):

Generic Receive Offload (IPv4 or IPv6)
Large Send Offload (IPv4 or IPv6)
Large Receive Offload (IPv4 or IPv6)
TCP Chimney (IPv4/IPv6)
TCP/UDP Checksum Offload (IPv4/IPv6)

In Windows based systems these may be displayed in:

>> '**Network Connections**' | **LAN** | **Properties** | **Configure** | **Advanced**:

These parameters may be disabled by going to

>> '**Control Panel**' | '**Network Connections**' | '**Local Area Network**' |
'**Properties**' | '**Configure**' | **Advanced** (or some variant of this)

and scrolling down the list in advanced and selecting '**disable**' or '**off**', then clicking **Ok**.

or

In Windows, go to **Control Panel** -> **Network and Internet Connections** -> **Network Connections**, right click the connection to change and choose '**Properties**'. Press the '**Configure...**' button, choose the '**Advanced**' tab to see or modify the "**Offload Transmit TCP Checksum**" and "**Offload Receive TCP Checksum**" values.

Also, the '**NETSH**' command line utility will also give information on the network capabilities of the installed operating system and installed network interface cards.

To see what IP/TCP/UDP options can be set on this OS, do this at the command prompt:

netsh int ip set /? <enter>

Below is the command to disable TCP Chimney if it is supported by the OS:

netsh int ip set chimney disabled <enter>

Virtual Environments

In addition to physical computer systems, many organizations and home users may choose to run products which allow the use of multiple operating systems on a single (physical) computer by use of Virtualization Software (VMWare, VirtualBox, Xen, etc).

These issues may still be present since the virtual environment is mostly dependent upon the physical hardware on which it is running. The above issues with LRO and GRO may not be present at the Virtual Machine level but rather on the 'host' operating system's physical hardware.

Additional Reference Material for Above Topics

TCP Offload Engine article on Wikipedia

hxxp://en.wikipedia.org/wiki/TCP_offload_engine

KB 912222, The Microsoft Windows Server 2003 Scalable Networking Pack Release

hxxp://support.microsoft.com/kb/912222

KB 951037, Information about the TCP Chimney Offload, Receive Side Scaling,
and Network Direct Memory Access features in Windows Server 2008

hxxp://support.microsoft.com/kb/951037

This issue can/may also exist in Windows Server 2003, 2008 and 2012.

Here are some URL's for additional information:

hxxp://blogs.msdn.com/b/psssql/archive/2010/02/21/tcp-offloading-again.aspx
hxxp://www.rackspace.com/knowledge_center/article/disabling-tcp-offloading-in-windows-server-2012
hxxp://social.technet.microsoft.com/Forums/windowsserver/en-US/3834ad00-94c1-4b78-8296-d917afa00111/server-2008-r2-file-copying-slow-when-recipient-iniaties-transfer?forum=winservergen

If you have any questions, comments or suggestions for improving this document, please feel free to send email to the address listed below.

Bill Parker
wp02855 at gmail dot com