Running Bro on BSD

An analysis of high performance solutions running on BSD operating systems.

Michael Shirk

BroCon 2016

@Shirkdog http://github.com/Shirkdog

Agenda

- Introduction
- Bro is awesome
- Why FreeBSD?
- High Performance and FreeBSD
- FreeBSD at the Berkley Lab
- PF_RING vs. netmap
- OpenBSD

Rant Warning

- Whenever you see the beastie with a hammer, there is a potential for some BSD bias to slip in.
- The goal is to minimize this throughout the talk.
- All information not cited in this talk is based on personal experience or opinion (marked with an asterisk *).

Introduction

- Worked in IDS/IPS since 2003 (various positions including consulting)
  - Engines: Snort, Suricata, Dragon and now Bro (also had to work with McAfee, ISS, NFR … others)
  - Signatures for Emerging Threats (since they were Bleeding Edge Snort)
- Support Open Source Security Tools and Software
  - Maintain pulledpork for Snort/Suricata (rule updating script): http://github.com/shirkdog/pulledpork
  - Active community member of open source projects:
    - Operating Systems: FreeBSD, OpenBSD, HardenedBSD
    - Security Tools: Snort, Suricata, AIDE, Bro (starting)

Bro Beginnings

- 2013 – Bro setup on Linux with PF_RING and Suricata (Dell R610 12 Core 32GB Appliance)
  - PoC was Security Onion, the production setup was on Ubuntu with PF_RING, Suricata and nothing else.
  - Gigamon TAP aggregated data to a single 10Gb Fiber interface fed to the Bro/Suricata sensor.
  - ~700Mbps peak, ~350Mbps non-peak
- Bro logs were fed into Splunk (modified Splunk_TA_Bro to work with log formats)
- Set it and forget it, it's just that simple.

Bro Beginnings

- Standalone install Bro 2.3
  - node.cfg

```
[bro]
type=standalone
host=localhost
interface=eth8
```

- Wait is this thing working correctly?
  - Setup capture-loss.bro
  - Reporting 40% loss.

Bro Beginnings

- Standalone Cluster setup with Bro 2.3 and PF_RING
  - node.cfg

```
[manager]
type=manager
host=localhost
[proxy-1]
type=proxy
host=localhost
[worker-1]
lb_method=pf_ring
lb_procs=5
type=worker
host=localhost
interface=eth8
```

- Cranked up lb_procs to 11 (really 10), Bro worked just fine, capture loss below 0.001%.

Bro Beginnings

- Intel Framework:
  - Setup with Critical Stack and other IOCs sourced elsewhere
- Upgraded to 2.4.0 (then to 2.4.1)
- The default install provided immediate value to analysts and aided in the incident response process.

Bro Epilogue

- After a number of years working with signature based IDS/IPS, Bro provided a great tool, where I could in theory passively monitor all of the things occurring on a network, and take either an active role, or perform further analysis at a later time with all available data.
- Essential to Network Security Monitoring (NSM)

Who uses Bro on Linux anyway?

- Ubuntu/Debian/RHEL/CentOS provide the necessary tools to run Bro on commodity hardware and network cards with PF_RING.
- Analysis tools (Splunk, ELK, etc.) mostly based on Linux platforms, integrate nicely with Bro.
- Security Onion – Runs on Ubuntu LTS and provides everything necessary to get Bro (and other tools) up and running easily.
- 99% use Linux *
  - Only the Berkley Labs and myself use it on BSD that I know of
  - I hope I am wrong :)

I do not run Linux on my systems anymore

- Ran Slackware Linux at first on my server, the only Linux I still like.
- Moved to OpenBSD for security on the server from about 2005 to 2009
- Moved to FreeBSD for support of some security tools (sguil/snorby) and network performance from 2009-2016
- Now run HardenedBSD (derivative of FreeBSD with ASLR and additional security features built-in).
- Ran Linux on my work laptop until 2012 then moved to pure FreeBSD for all work related items (except a MacBook).
- Everything I care about runs FreeBSD/HardenedBSD and ZFS.

Shellshock, systemd, oh my!

- <rant>Default shells in BSD are not bash
  - Wait, I need bash for Bro to run…
  - Its okay, /bin/sh is not linked to /bin/bash, bash is in /usr/local/bin/bash where it belongs!
- Comparison of lines of code:
  - FreeBSD init.c = 1989
  - systemd = ~480,000 (from openhub.net)
- More lines of code == more vulnerabilities
- OS Diversity is important – For George</rant>

Why FreeBSD?

- Simplistic:
  - Small install footprint, UNIX-Like operating system with ports/pkgs for everything you need, and only what you add.
- Performance *:
  - FreeBSD 10 was one of the best performing operating systems I had ever used.
  - Specific workloads require specific tuning, like any operating system, but pushing my BSD laptop to its limits (running current code) was more stable and reliable than my MacBook (Virtualization/ZFS)
- History:
  - FreeBSD is derived from the original AT&T Unix (Lehy, 2016). The codebase has evolved over the past 40 years as a high performance server operating system.
- Community:
  - A great community of developers and users
  - FreeBSD Foundation provides support to the community and conferences.
- Security *:
  - Out of the box FreeBSD provides a secure base operating system without unnecessary services and software, with some additional tweaks to further harden the operating system.
  - HardenedBSD adds in a number of security features into the FreeBSD base installation to be resilient to exploitation of software vulnerabilities.

Why FreeBSD continued

- bhyve:
  - Type-2 hypervisor installed by default with FreeBSD 10.0+
  - Easy to setup VMs for FreeBSD, OpenBSD, Linux and even Windows. (Graphical support coming in FreeBSD 11).
- ZFS:
  - Data-integrity at its core
  - Copy-on-write features, including snapshots making backups simple.
- Jails
  - Where containers came from (well Solaris Zones from Jails) , service based or full OS jails.
  - Simple to test software in a jail, and in some cases, run applications within their own jail. (Possible with Snort/Suricata and Bro?!?)
- These three features I use for all of my work.

Why others use FreeBSD?

- Netflix:
  - “FreeBSD was selected for its balance of stability and features, a strong development community and staff expertise.” (Netflix, 2016).
  - 32% of North American Internet traffic is served by FreeBSD using their OpenConnect Appliance (FreeBSD, 2016).
- Juniper:
  - JunOS is based on FreeBSD, and powers all Juniper network equipment (FreeBSD, 2016).
- WhatsApp:
  - Their platform running on FreeBSD can support 2.5 Million concurrent connections per server (FreeBSD, 2016).
- NetApp, as well as others (See FreeBSD link in references more)

How to setup Bro on FreeBSD

- Bro 2.4.1 is in the ports tree (FreeBSD 10/11) and exists as a binary package.
- After an installation of FreeBSD with networking setup, the following command will install Bro:

```
pkg install -y bro
```

```
cd /usr/ports/security/bro && make install clean
```

- If the ports tree was installed during the installation of FreeBSD, you can install with the port, allowing changes to be made to the configuration. The following will install Bro by compiling it with all of the necessary dependencies:

How to setup Bro on FreeBSD

- The port and package has a prefix of /usr/local, so Bro files are installed in /usr/local/etc, /usr/local/share/bro/site
- rc script for Bro?
  - Seems to be missing in the package (Bug submitted)
  - Easy enough to make, or add the following to /etc/rc.local

```
/usr/local/bin/broctl deploy
```

How I setup Bro on FreeBSD

- Source code all the way (use gmake)

```
fetch https://www.bro.org/downloads/bro-2.4.1.tar.gz
tar xzf bro* && cd bro-2.4.1
./configure -prefix=/opt/bro2 \ --
logdir=/nsm/bro2/logs \
--spooldir=/nsm/bro2/spool
gmake
sudo gmake install
```

- Disable checksums, offloading on your interface of choice:

```
/sbin/ifconfig vtnet0 -rxcsum -rxcsum6 -txcsum -txcsum6 -tso -lro
```

- You need the source to setup the netmap plugin (described later, and an item that I need to get fixed in the FreeBSD port/pkg).

How I setup Bro on FreeBSD

- Another option is to setup Hunter-NSM, available at: http://github.com/Shirkdog/hunter-nsm
- Hunter-NSM is a simple install script for Snort/Bro IDS with JSON logging on FreeBSD
- Suricata support is coming soon, with some updates to the ids-tools for JSON output for unified2 output.

High Performance and FreeBSD

- Though PF_RING is tied to the Linux kernel, there are several options that can be used to boost performance of Bro on FreeBSD:
  - FreeBSD tuning (utilize queues)
  - Chelsio/Myricom (specialized network cards for packet processing)
  - Netmap (netmap framework)
  - Packet-bricks/LB (Based on the netmap framework)

FreeBSD Tuning

- Best website for open source configurations specifically OpenBSD/FreeBSD:

  https://calomel.org/freebsd_network_tuning.html

- Intel cards as the choice on commodity hardware (normally the best driver support)
- NIC queues correlate to the available cores of the CPU

FreeBSD Tuning

- Intel igb driver tuning, updates are made to /boot/loader.conf on FreeBSD.
- "Testing has shown a server can most efficiently process the number of network queues equal to the total number of CPU cores in the machine" (Calomel, 2016).

```
hw.igb.num_queues="2" # (default 0)
```

- In this example, a dual port Intel I-350 on a machine with 4 CPU cores should use 2 network queues per port.

FreeBSD Tuning

- Network queues can be utilized with Bro
- Available lb_methods: pf_ring, myricom, custom, or interfaces.
- Example with two interfaces in a worker node.

```
[worker-1]
type=worker
host=localhost
lb_method=interfaces
lb_procs=2
lb_interfaces=igb0,igb1
```

FreeBSD Tuning

- Method requires a front-end to process packets
  - Load balancing
  - Aggregation
  - Hashing
- Still using libpcap

Chelsio NICs

- High performance 1Gb/10Gb/40Gb NICs
- Well supported drivers within FreeBSD (cxgb,cxgbe)
- Similar tunables available to maximize receive performance.

Myricom NICs

- High performance 10Gb NICs
- Standard driver support within FreeBSD (mxge)
- Sniffer10G
  - Licensed Add-On feature, enabling lossless packet capture for a variety of open source tools (Bro, Snort, Suricata, Wireshark/tcpdump)
- Plugin available for Bro.

Myricom NICs

- Example of a worker node configuration with a Myricom card + Sniffer10G license:

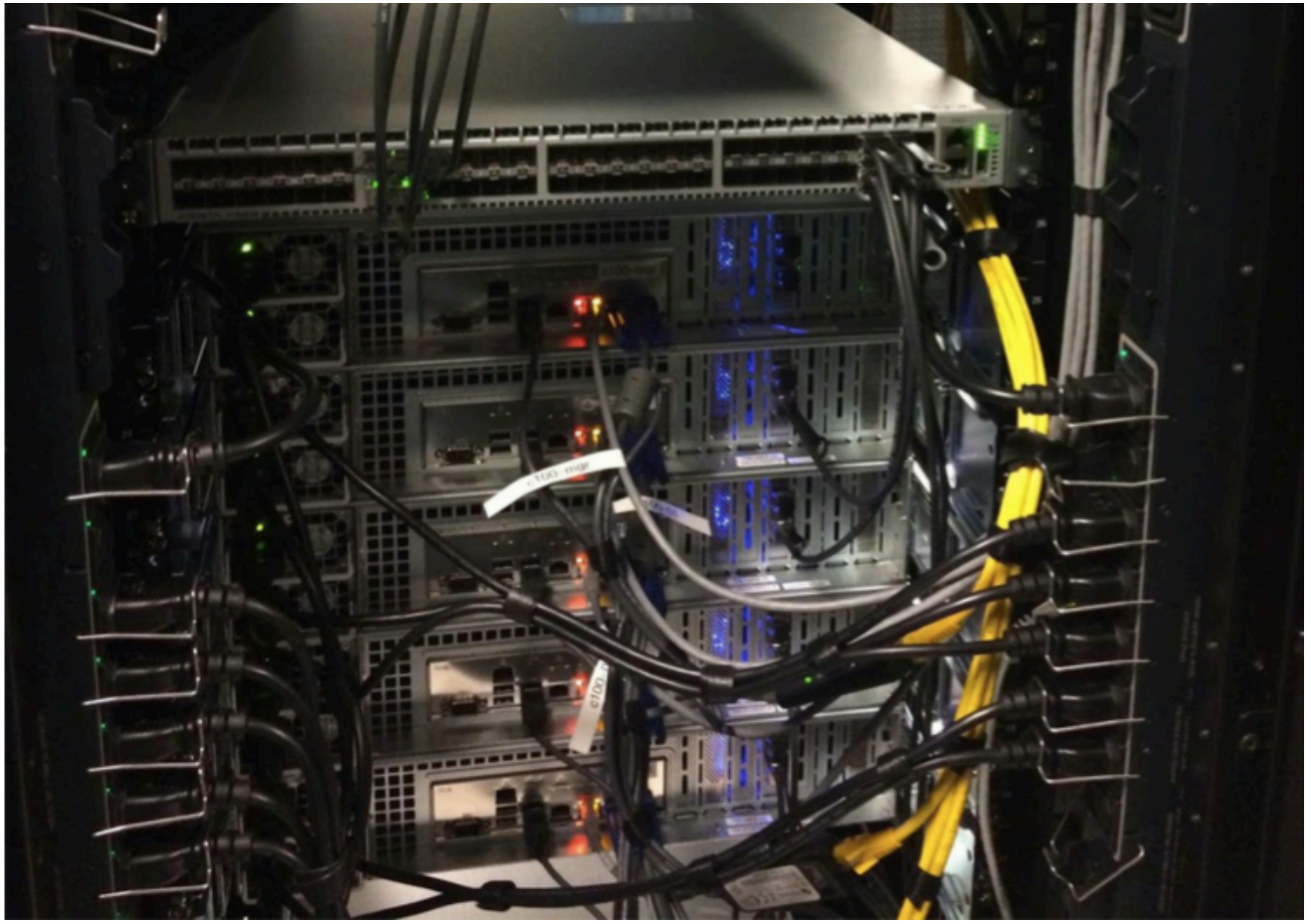
```
[worker1]
type=worker
host=LBL-Worker-Host
interface=myri0
lb_method=myricom
lb_procs=10
pin_cpus=3,5,7,9,11,13,15,17,19,21
env_vars=LD_LIBRARY_PATH=/usr/local/opt/snf/lib:$PATH,
SNF_DATARING_SIZE=0x100000000,
SNF_NUM_RINGS=10,SNF_FLAGS=0x1
```

- Advanced features for pinning specific CPUs, and setting other settings specific to Myricom.

Myricom NICs at the Berkley Lab

- “We evaluated several options for host distribution, our primary requirement being that it would support our preferred OS of FreeBSD” (Stoffer, Sharma, & Krous, 2015).
- Bro cluster with FreeBSD servers, with Arista switches as the traffic distribution solution.
  - 5 Servers with FreeBSD, each with 1 proxy node, and 10 worker nodes (1 server runs the manager node)
  - Myricom card splits the 10Gb traffic fed to each server into 1Gb streams for each Bro Process
- See the references section for the complete document, which has all of the configurations used for the configuration of all of the technology.

FreeBSD and Bro at the Berkley Lab



Source: (Stoffer et al., 2015)

Myricom NICs at the Berkley Lab

- Two pieces of technology that were evaluated for the 100Gb solution for traffic distribution were netmap and packet-bricks (Stoffer, Sharma, & Krous, 2015).

netmap framework

- Framework for fast packet processing, utilizing a userspace API to access raw packets from network devices (Rizzo, 2012).
- Works on Linux and FreeBSD
  - In FreeBSD 11 netmap will be enabled by default in the kernel
- Allows for userspace applications to access raw packets by detaching the network card from the host stack, giving access directly to packets in the network buffers.
- Caveat: The netmap enabled interface will no longer be receiving or transmitting packets through the kernel and normal protocol subsystems, placing it into a “netmap mode”

netmap framework

- netmap enabled applications handle the moving of packets on the mmaped buffers (rings)
  - The application can process the packets, or forward the packets onto the kernel subsystems
- netmap pipes provide dynamic access to packet rings
  - All memory is shared by network interfaces in netmap mode, where netmap pipes use separate memory segments by default.

netmap framework

- On FreeBSD 11+, interfaces can be dynamically accessed as netmap interfaces

```
tcpdump -i netmap:em0 -nns 0
```

- Other features include pure virtual networking with VALE switches
  - netmap provides a mechanism for the forwarding of packets at line rate between virtual ports.

netmap and Bro

- Of course there is a Bro plugin that supports netmap mode, but it has to be built to be available
- The following must be run in $SRCDIR/aux/plugins/netmap to enable the plugin
- FreeBSD src or netmap source code must be passed to the “--with-netmap” flag

```
./configure --bro-dist=/usr/src/bro-2.4.1 \
--install-root=/opt/bro2/lib/bro/plugins \
--with-netmap=/usr/src
```

netmap and Bro

- Syntax is the same for standalone/worker nodes for using netmap plugin just the interface is different (currently...info later).
- Run the following to verify the netmap plugin is available to Bro

```
/opt/bro2/bin/bro -N Bro::Netmap
```

- Add the following to your worker configuration in node.cfg

```
[worker-1]
type=worker
Host=192.168.2.5
interface=netmap::igb1
```

netmap + packet-bricks and Bro

- packet-bricks is a netmap-based packet layer for distributing and filtering traffic: https://github.com/bro/packet-bricks
- Provides the necessary front-end to netmap by way of netmap pipes
- C and Lua: Scripts can be written in Lua to program functionality.
- Updated to provide load balancer script in the default install (thanks Asim).
- The following will setup a load balancer with igb1, and setting up 4 netmap pipe channels:

```
/opt/packet-bricks/bin/bricks-load-balance igb1 4
```

netmap + packet-bricks and Bro

- Slightly different syntax for the interfaces when setup on multiple workers (for netmap pipes):

```
[worker-1]
type=worker
host=192.168.2.5
interface=netmap:igb1}0

[worker-2]
type=worker
host=192.168.2.5
interface=netmap:igb1}1
```

Test Setup

- Comparision of the same hardware with PF_RING on CentOS 7 vs. netmap on FreeBSD
- Commodity Hardware:
- CPU: Intel(R) Core(TM) i3-3225 CPU @ 3.30GHz (3300.09-MHz K8-class CPU)
- Memory: 8GB
- NIC: Intel I350-T2 Dual Port (one port used for monitoring)
- Sender Source: HardenedBSD (FreeBSD) 12-Current with tcpreplay+netmap

Test Setup

- Traffic:
  - Enterprise mix pcap with tons of garbage:

  conn.log, dhcp.log, dns.log, dpd.log, files.log, ftp.log, http.log, kerberos.log, modbus.log, mysql.log, notice.log, packet_filter.log, rdp.log, smtp.log, snmp.log, socks.log, ssl.log, syslog.log, traceroute.log, weird.log, x509.log
- Rate:
  - 500Mbps
  - 150,000pps
- Limitation: Hardware is not ideal, but provides loss to measure the performance improvements of different configurations on FreeBSD

Test Setup

- Linux:
  - CentOS 7
  - PF_RING (git repo current)
  - Bro 2.4.1 (src download)
- FreeBSD:
  - HardenedBSD (FreeBSD 12-CURRENT)
  - netmap (current code in FreeBSD tree)
  - packet-bricks (git repo current)
  - Bro 2.4.1 (src download)

Test Setup

- FreeBSD Tunables:

/boot/loader.conf

```
net.link.ifqmaxlen="2048"
hw.igb.txd=2048
hw.igb.rxd=2048
hw.igb.num_queues="4"
hw.igb.rx_process_limit="-1"
```

/etc/rc.local

```
/sbin/ifconfig igb1 -rxcsum -rxcsum6 -txcsum -txcsum6 -tso
-lro promisc up
```

Test Results

OS	NodeConfig	Capture-loss	netmap	comments
CentOS	standalone	43.5	no	none
CentOS	Cluster2	18.5	no	Two lb procs
CentOS	Cluster5	17.1	no	Four lb procs
FreeBSD	standalone	38	no	none
FreeBSD	standalone	37	yes	Raw without plugin
FreeBSD	Cluster2	20.55	yes	Packet-bricks and two workers
FreeBSD	Cluster4	13.13	yes	Packet-bricks and four workers
FreeBSD	Cluster4	16.56	yes	Packet-bricks and four workers

Test Results

- Results were similar between PF_RING on Linux and netmap on FreeBSD with the same hardware, traffic mix and Bro code.
- The results mimic other research into the various packet capture solutions, such as PF_RING, netmap and DPDK, mainly that netmap performs well for being a adaptable framework for packet capture at the cost of some performance (Gallenmuller, 2014).

Test Observations

- Some issues were observed with broctl on FreeBSD when trying to stop and start with new configuration and netmap pipes.
  - This might have been an error on my part, path issue with broctl, but it did not happen all the time.
- Bro seems to get upset if netmap interfaces are not setup correctly.
- Additional testing is required to gather more data.

netmap, FreeBSD and Bro 2.5

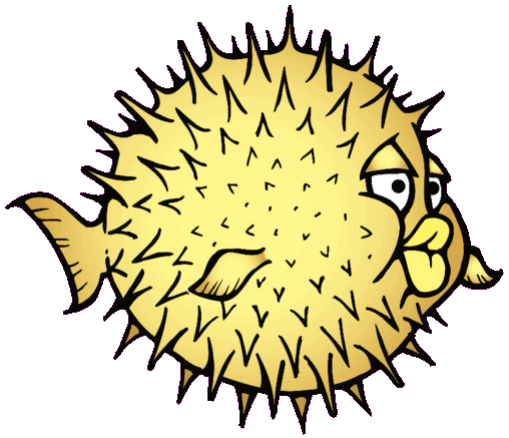
- Information on github has been updated to reflect a new syntax for the netmap plugin
  - Looks to be similar to how PF_RING is configured on Linux
- Additional reference to LB application provided in the netmap github repo
  - This is not available in the FreeBSD, however, the LB app can be built and appears to work fine in FreeBSD current code.
- Working with the FreeBSD developers to get the netmap code in FreeBSD in sync with the current code on github
  - Enhancements and new features like LB.

OpenBSD?!?

- I had something about OpenBSD in the agenda.

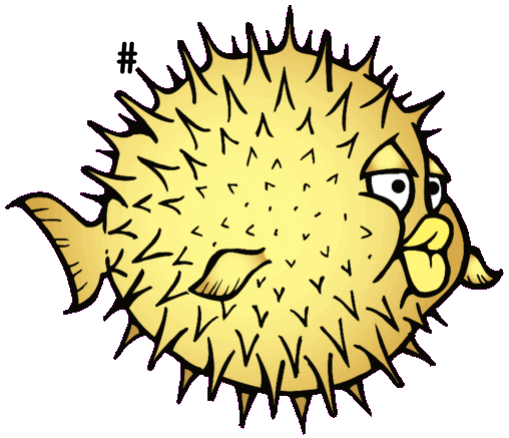
Bro on OpenBSD 6.1

- The OpenBSD Port has been updated in OpenBSD current, and will arrive in OpenBSD 6.1 in 2017
- Open file limits have to be adjusted for broctl to work (documented in the README)
- Interface is set to the default of eth0 and must be changed (Linux names do not belong here)
- Thanks to Stuart and Antoine from the OpenBSD Project for resolving the issues with the port.



Bro on OpenBSD 6.1

```
# uname -sr
OpenBSD 6.0
# /usr/local/bin/broctl start
starting bro ...
# /usr/local/bin/broctl status
Getting process status ...
Getting peer status ...
Name         Type       Host          Status    Pid    Peers   Started
bro          standalone localhost     running   68509  ???     10 Sep
   06:57:14
#
```



Conclusion

- There are alternatives available for those that want to use another operating system other than Linux without sacrificing the performance of tools like PF_RING.
- Several solutions have been discussed for FreeBSD, which vary from freely available to a cost for increasing performance.
- FreeBSD provides a great platform for running Bro, and now OpenBSD *

References

- Calomel. (2016). FreeBSD tuning and optimization. Retrieved from https://calomel.org/freebsd_network_tuning.html
- FreeBSD. (2016). Welcome to FreeBSD. Retrieved from https://www.freebsd.org/doc/handbook/nutshell.html
- Gallenmuller, S. (2014). Comparison of memory mapping techniques for high-speed packet processing. Master's thesis, *Technische Universitat Munchen,* Germany.
- Lehey, G. (2016). Explaining BSD. Retrievedfrom https://www.freebsd.org/doc/en/articles/explaining-bsd/article.html
- Netflix. (2016). Appliance software, OpenConnect. Retrieved from https://openconnect.netflix.com/en/software/
- Rizzo, L. (2012). Netmap: a novel framework for fast packet I/O. *2012 USENIX Annual Technical Conference.*
- Stoffer, V., Sharma, A., & Krous, J. (2015). 100G Intrusion Detection. Berkeley Lab. Retrieved from *http://commons.lbl.gov/download/attachments/120063098/100GIntrusionDetection.pdf*

Questions?