ARISTA Arista EOS™ integrated traffic visualization with Splunk

Arista Solution Brief – January 2013

Preface

The need for real time traffic information is becoming a growing requirement within a majority of data centers today. Source and destination information, top talkers, top web sites, packet discards, security checks, and packet latency information are being requested, on demand, from network, application and security administrators. The changes in traffic patterns, the sheer volume of traffic and the exponential increase in the data rates are making it difficult to capture, analyze, and troubleshoot traffic, as well as to provide meaningful reports to the application and business constituents.

Adding to these challenges is the fundamental shift in the design of data center networks where the predominant traffic no longer passes through centralized north/south aggregation boundaries points. Traffic is becoming far more distributed and multi-directional as driven by data center virtualization, leaf spine architectures, and high performance like computing clusters with server-to-server, server-to-storage, and virtualized application appliances (L4-L7 services). In addition, given the exponential increases in traffic forwarding rates, where 10GbE and 40GbE are becoming the norm and port-to-port switching latency in the sub microsecond range, there is a need to capture traffic data at much higher data rates and with greater accuracy often with time stamps accurate in the low teen nanoseconds.

Unfortunately, the existing centralized probe architecture, and/or centralized network analysis module architectures do not transition well to these fundamental changes. Customers must re-architect what has been a significant investment in network analysis tools (probes, specialized analysis modules) and proprietary software to analyze the captured data and provide reports. Fortunately, there are new approaches to traffic monitoring and analysis, including line rate traffic intelligence within the switches, and open source analysis and reporting tools, that require much lower investments as customers rethink and redesign their traffic capture and analysis architectures.

This Arista Solution Brief discusses a new approach to network traffic monitoring on Arista switches.

Arista Company Background

Arista Networks delivers a portfolio of 1/10/40 and 100 GbE capable switching platforms that redefine network architectures, bring extensibility to networking, and dramatically change the price/performance of data center networks. At the core of Arista's platform is the Extensible Operating System (EOS™), a groundbreaking network operating system with single-image consistency across both fixed and modular chassis, and a modern core architecture enabling in-service upgrades, customer created Linux utilities, and application extensibility.

Arista EOS provides ways to gain visibility into network traffic, mirror or steer, and supports triggers/actions based on events but provides no historical reporting/searching or graphical visualization. The extensibility of EOS, with off-the-shelf software from Splunk, provides this. By leveraging Splunk within their network operations, administrators can have real time traffic visibility, reporting and visualization from Arista switching platforms. This is offered without the need for costly hardware switch upgrades or specialized traffic analysis modules. Moreover, this highlights the benefits of an open and extensible platform on Arista switches.

Solution Overview

Splunk Enterprise (http://www.splunk.com/) is software that processes machine-generated data in real-time and provides real-time visibility, insight and intelligence via powerful search capabilities.

Splunk ingests machine-generated data from data center infrastructure platforms including server, networking, and storage devices. To date many Splunk use cases have been server application and storage centric or simply based on control-plane log (e.g. syslog) output. This solution brief demonstrates the use of Splunk for capturing distributed traffic data from Arista switches via sFlow and with simple search routines, shows how to generate real time reports on traffic behaviors.

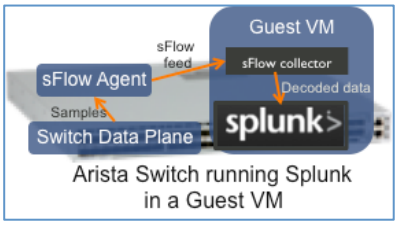
Arista Solution Brief – January 2013

This solution involves taking samples of data-plane traffic, analyzing the fields of the sampled data, ingesting that into Splunk software running on an Arista switch (making use of the extensibility in EOS) and providing an integrated solution where sampled traffic is combined with Splunk's extensive reporting and visualization tools to provide visibility into traffic in real-time, on the switch directly, without any need for external servers or storage.

This solution utilizes on a number of underlying foundation technologies:

- **EOS extensibility** allows installation of 3rd party RPMs directly into Linux on the control-plane and 3rd party code to operate within a **Guest Virtual Machine (VM)** on the switch itself
  - The **decoupling of data-plane forwarding** (silicon) from control-plane (software) enables deploying applications on the switch with no impact on network performance
  - The **x86 control-plane** of Arista switches (multi-core x86 Xeon-class CPU, many gigabytes of RAM) running atop Linux enabling 3rd party software to be installed as-is without modification
- Arista switches optionally ship with an **Enterprise grade solid state disk (SSD)** for additional persistent storage and EOS extensibility can be used to access 3rd party storage via NFS, CIFS etc.
- Arista switches provide **scripting and Linux** (bash) shell level access for automation
- Splunk Enterprise software is provided as a Linux RPM that can be installed directly on Arista switches

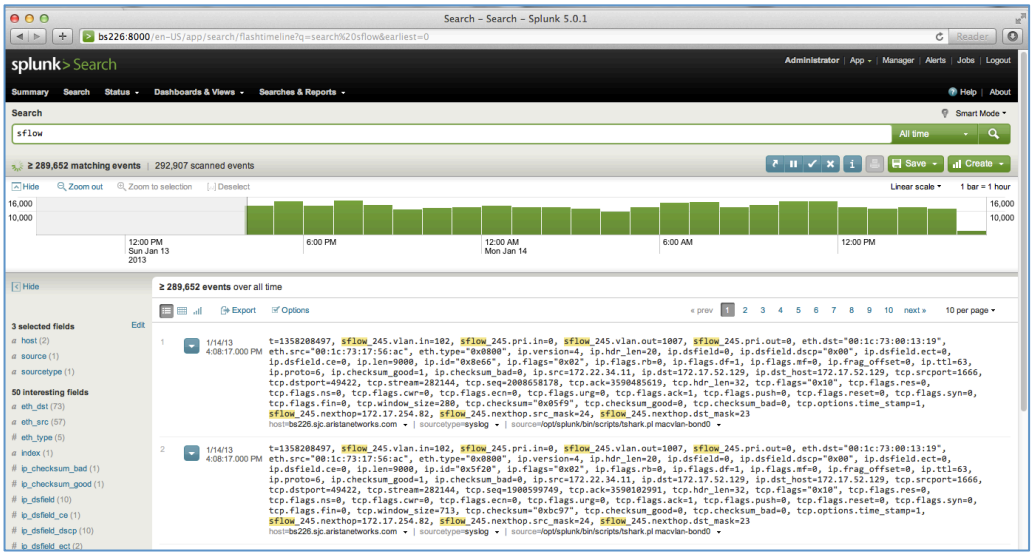
Many EOS extensions can run on the same Linux image as EOS itself, however as Splunk can have quite large resource demands a Guest VM has been chosen for this solution brief. Splunk has been deployed as an OpenStack Fedora Linux guest VM image. This provides an optimal balance between flexibility and extensibility while also ensuring separation of applications for reliability and resiliency.



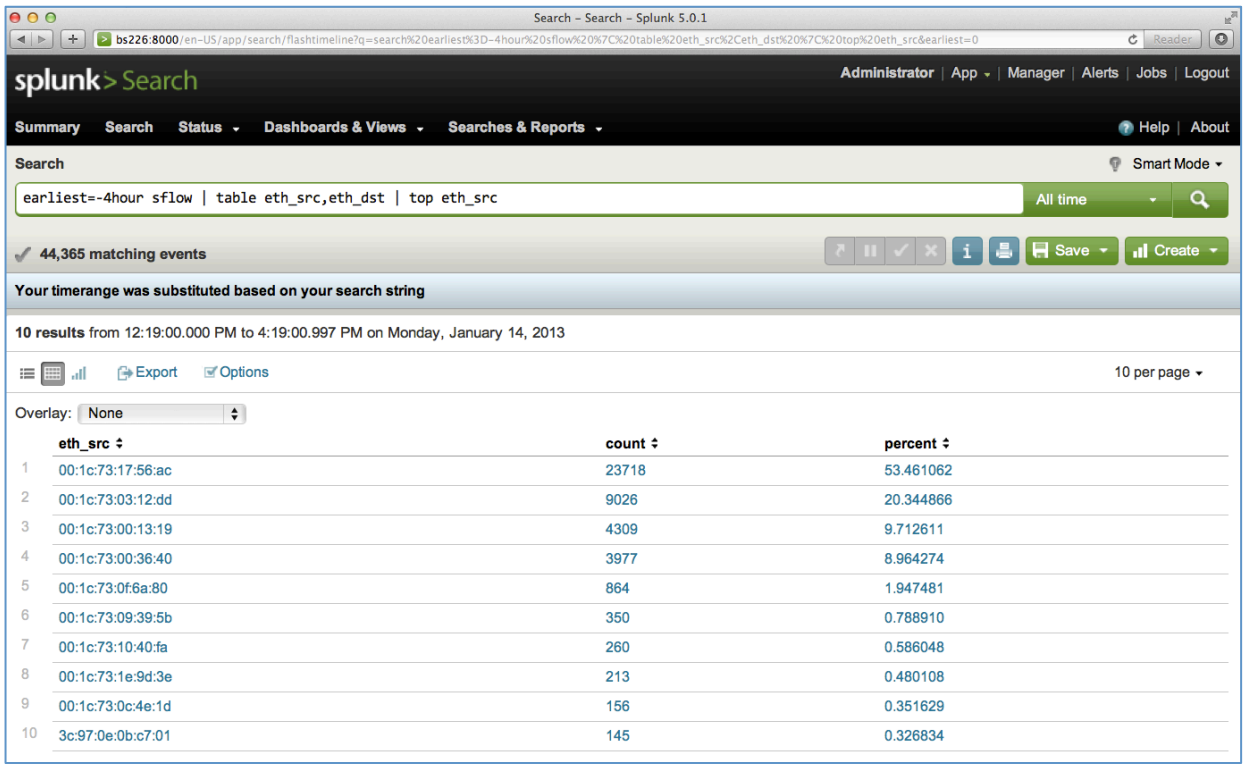
Splunk Output, Search Queries and Reports

Ingesting decoded sFlow sampled data into Splunk enables search queries based on frame/packet fields. Since sFlow provides the first 128 bytes of each sampled packet, there is not just L2-L4 fields but also the start of packet payloads and things like HTTP, SIP and other protocol headers ready for searching.

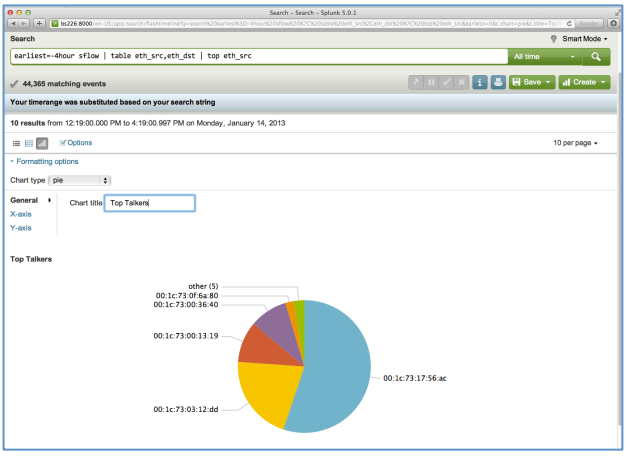
The display below from Splunk's web interface shows frames/packets streaming into Splunk with it waiting for search terms based on field headers:

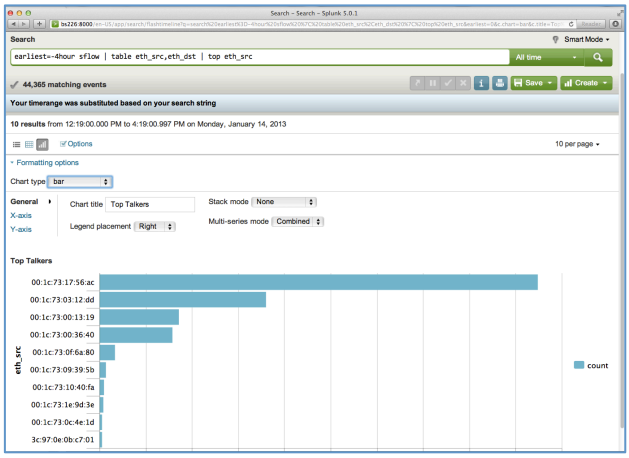


Splunk supports advanced ad-hoc queries of real-time data. For example, if there is more traffic within a VLAN than what is projected this could be analyzed by constructing a search query for 'Top Talkers' in a given VLAN at layer 2 (eth_src, eth_dst fields) in the last 4 hours. Such a query could be "**earliest=-4hour sflow | table eth_src,eth_dst | top eth_src**" with the results displayed as a table or graph as shown below:

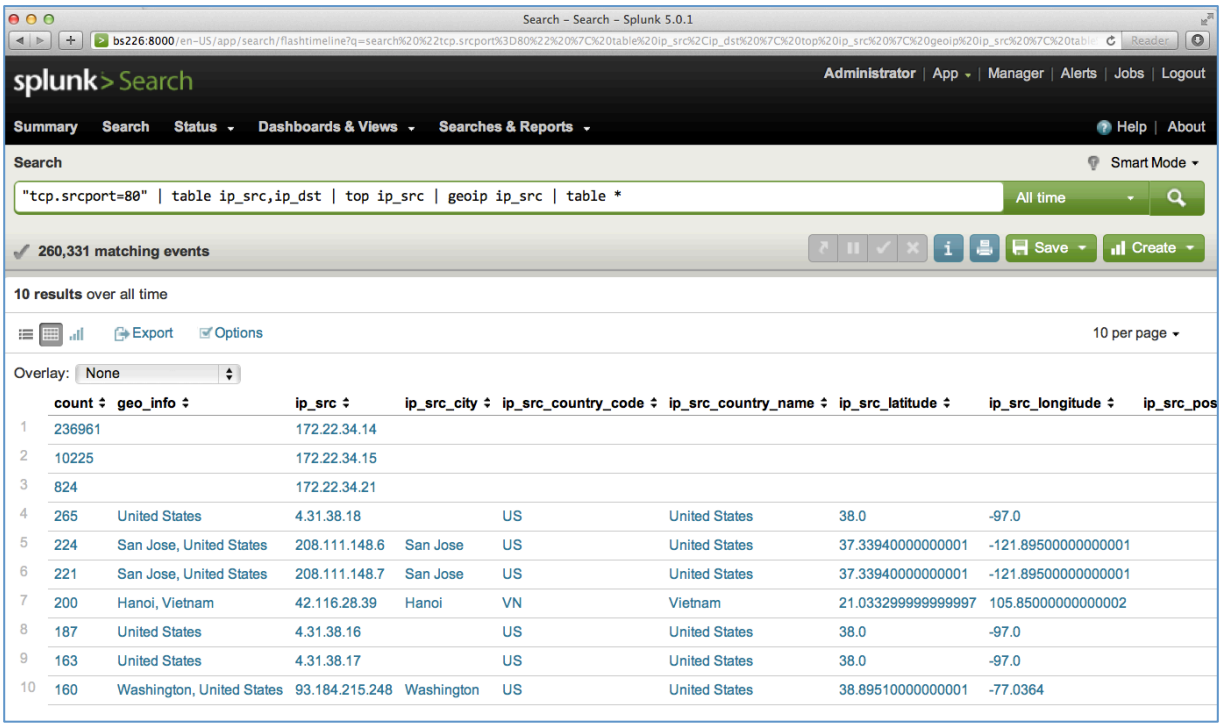


	eth_src	count	percent
1	00:1c:73:17:56:ac	23718	53.461062
2	00:1c:73:03:12:dd	9026	20.344866
3	00:1c:73:00:13:19	4309	9.712611
4	00:1c:73:00:36:40	3977	8.964274
5	00:1c:73:0f:6a:80	864	1.947481
6	00:1c:73:09:39:5b	350	0.788910
7	00:1c:73:10:40:fa	260	0.586048
8	00:1c:73:1e:9d:3e	213	0.480108
9	00:1c:73:0c:4e:1d	156	0.351629
10	3c:97:0e:0b:c7:01	145	0.326834



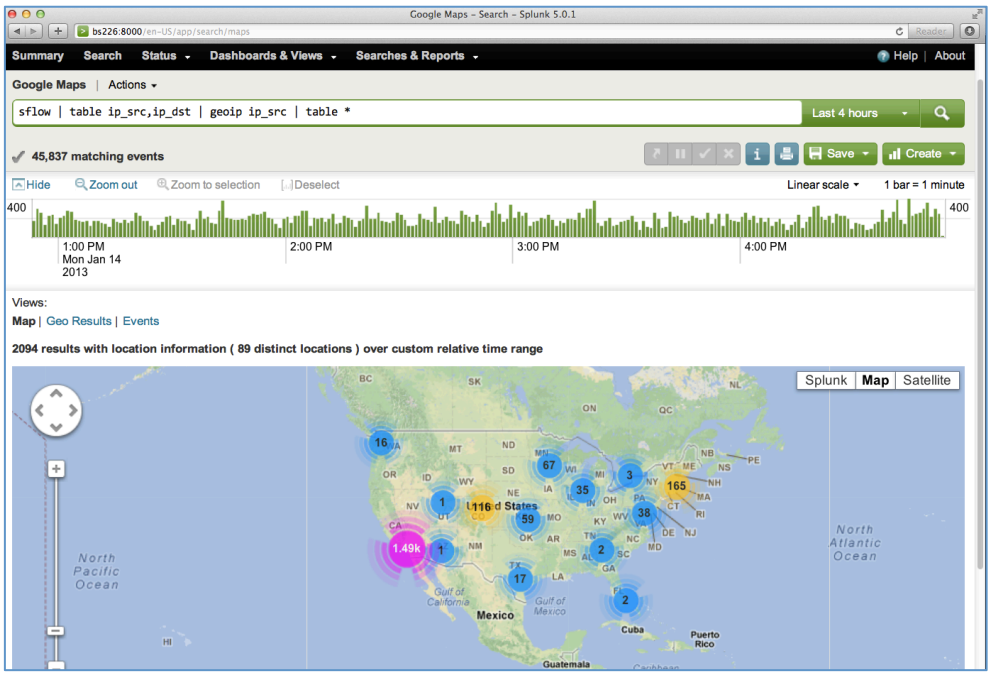


Splunk can post-process things like IP address fields and augment it with additional information such as perform a geo-location lookup. For example, to perform analysis on what countries you have most HTTP traffic originating from using a query in Splunk such as "**tcp.srcport=80 | table ip_src,ip_dst | top ip_src | geoip ip_src | table ***":

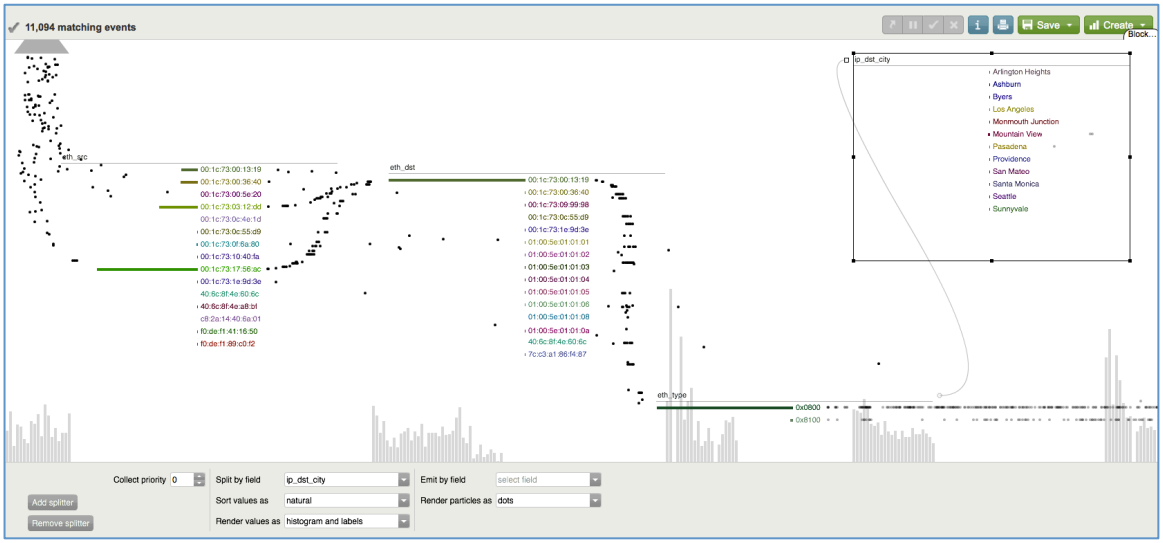


	count	geo_info	ip_src	ip_src_city	ip_src_country_code	ip_src_country_name	ip_src_latitude	ip_src_longitude	ip_src_pos
1	236961		172.22.34.14						
2	10225		172.22.34.15						
3	824		172.22.34.21						
4	265	United States	4.31.38.18		US	United States	38.0	-97.0	
5	224	San Jose, United States	208.111.148.6	San Jose	US	United States	37.33940000000001	-121.89500000000001	
6	221	San Jose, United States	208.111.148.7	San Jose	US	United States	37.33940000000001	-121.89500000000001	
7	200	Hanoi, Vietnam	42.116.28.39	Hanoi	VN	Vietnam	21.033299999999997	105.85000000000002	
8	187	United States	4.31.38.16		US	United States	38.0	-97.0	
9	163	United States	4.31.38.17		US	United States	38.0	-97.0	
10	160	Washington, United States	93.184.215.248	Washington	US	United States	38.89510000000001	-77.0364	

Things like geolocation data can be overlayed on to a map. For example, to show where traffic is originating from a search query such as "**sflow | table ip_src,ip_dst | geoip ip_src | table ***" with the 'view' set to Google Maps produces results like below:



Splunk Particle provides additional advanced real-time visualization and tools for looking at sFlow sampled traffic and this can be used for real time data forensics. The video at http://www.youtube.com/watch?v=-q-Ue603vKc shows this in action, below is a screenshot of Splunk Particle showing live traffic streaming in, grouped by eth_src initially then eth_dst, eth_type and finally ip_dst_city for any packets that resolve to a geolocation:



There is almost no limit to the possible queries and use cases:

- Application performance and SLA monitoring
- Both packet and flow based traffic analysis, recording and monitoring
- Real time and historical Analytics
- Standard API access to packet capture data
- Access from anywhere

Configuration Recipe: Switch setup

Arista's Extensible Operating System (EOS) provides best-in-class flexibility in terms of how switch functions can be enhanced and extended. EOS is easily extended by loading a Linux Guest VM image; Splunk is then installed in-top-of this Linux Guest VM..

To start with, a Guest VM instance of Linux is required. A relatively easy way of getting an image to start with is to use a Fedora JEOS (Just Enough OS) pre-built image as used within OpenStack.

The page at http://docs.openstack.org/trunk/openstack-compute/admin/content/starting-images.html provides a number of links to pre-built images, in this paper Arista is using the f16-x86_64-openstack-sda.qcow2 image.

Copy the image to an Arista switch and provision the Guest VM as follows. Note that the Guest VM image is installing in the 'drive:' filesystem as this switch has the (optional) enterprise-grade SSD installed. If that is not the case then 'flash:' filesystem or other could be utilized instead.

```
switch> enable
switch# copy http://berrange.fedorapeople.org/images/2012-02-29/f16-x86_64-openstack-sda.qcow2 drive:
switch#
```

Once the image is installed on the switch a configuration is then added. While many of these parameters can be set via 'virtual-machine' configuration directives, Arista is making some non-standard host-to-guest VM connectivity that requires a customized XML setup.

You can follow the instructions on the EOS Central article about creating Guest VM instances on EOS at https://eos.aristanetworks.com/wiki/index.php/Configuration:VM however for the purposes of this paper it is suggested to simply make use of the preconfigured Guest VM configuration file.

This can be downloaded from EOS Central with:

- Switch with integrated Enterprise grade SSD:

```
switch# bash

Arista Networks EOS shell

[admin@switch ~]$ sudo wget 'https://eos.aristanetworks.com/wiki/index.php?action=raw&title=Configuration:VM:guest_vm_fc16_ssd' -O
/mnt/flash/foo.xml
Resolving eos.aristanetworks.com... 50.19.101.37
Connecting to eos.aristanetworks.com|50.19.101.37|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2213 (2.2K) [text/x-wiki]
Saving to: `/mnt/flash/foo55.xml'

100%[======================================================================================>] 2,213       --.-K/s   in 0s
2013-01-15 06:01:01 (27.6 MB/s) - `/mnt/flash/foo.xml' saved [2213/2213]
[admin@switch ~]$ exit
switch#
```

- Switch without SSD:

```
switch# bash

Arista Networks EOS shell

[admin@switch ~]$ sudo wget 'https://eos.aristanetworks.com/wiki/index.php?action=raw&title=Configuration:VM:guest_vm_fc16_no_ssd' -O
/mnt/flash/foo.xml
Resolving eos.aristanetworks.com... 50.19.101.37
Connecting to eos.aristanetworks.com|50.19.101.37|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2213 (2.2K) [text/x-wiki]
Saving to: `/mnt/flash/foo55.xml'

100%[======================================================================================>] 2,213       --.-K/s   in 0s
2013-01-15 06:01:01 (27.6 MB/s) - `/mnt/flash/foo.xml' saved [2213/2213]
[admin@switch ~]$ exit
switch#
```

Next, instantiate the Guest VM via the following configuration:

```
switch# config terminal
switch(config)# virtual-machine foo                    Name of guest VM
switch(config-vm-foo)# config-file drive:foo.xml       VM XML Configuration
switch(config-vm-foo)# enable                          Start it running
switch(config-vm-foo)# exit
switch(config)# exit
switch#
```

At this point the Guest VM instance should be running and one can use a VNC client to the switch on port 5900 to connect to its console. The control-plane ACL on EOS by default allows port 5900 through.

Configure an event-handler on EOS to setup a point-to-point IP connection on the 'br0' bridge interface for sFlow traffic to get to the Guest VM:

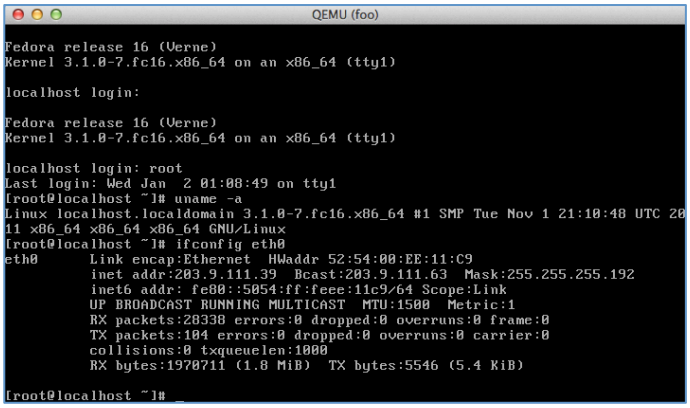
```
switch# config terminal
switch(config)# event-handler foo
switch(config-handler-foo)# trigger onBoot
switch(config-handler-foo)# delay 30
switch(config-handler-foo)# action bash sudo ifconfig br0 192.168.254.1/24
switch(config-handler-foo)# exit
switch(config)# exit
switch# bash sudo ifconfig br0 192.168.254.1/24
switch#
```

Enable sFlow on the switch with traffic sourced from whatever interfaces and a destination that points out the br0 interface (e.g. 192.168.254.2/24 as the guest VM). An example below shows sFlow enabled sampling 1 in 20,000 packets:

```
switch# config terminal
switch(config)# sflow sample 20000                     Sample 1 in 20K packets
switch(config)# sflow polling-interval 3600
switch(config)# sflow destination 192.168.254.2        Send sFlow packets at the Guest VM br0 interface
switch(config)# sflow source 10.1.1.1                  Originate sFlow from our 10.1.1.1 IP address (customize as appropriate)
switch(config)# sflow run                              Enable sFlow
```

Configuration Recipe: Guest VM / Splunk setup

With the Guest VM running on the switch, use a VNC client/console application to log into the Guest VM



Guest VM VNC Console

Within the Guest VM configure eth1 to connect to the switch and validate it is alive:

```
[root@localhost ~]# ifconfig eth1 192.168.254.2/24
[root@localhost ~]# ping 192.168.254.1
```

Install Splunk Enterprise software. As Splunk software is available as a RPM its simply a case of installing the RPM:

```
[root@localhost ~]# rpm –i splunk-5.0.1-143156.i386.rpm
-----------------------------------------------------------------------
Splunk has been installed in:
        /opt/splunk
To start Splunk, run the command:
        /opt/splunk/bin/splunk start
To use the Splunk Web interface, point your browser at:
    http://x.x.x.x:8000
Complete documentation is at http://docs.splunk.com/Documentation/Splunk
-----------------------------------------------------------------------
[root@localhost ~]# /opt/splunk/bin/splunk start
```

Splunk itself doesn't understand sFlow packet sampling data, so a script needs to be installed which can take sFlow traffic, decode it into a format that Splunk can parse and index and use that as a data-source.

We have a sample script on EOS Central that provides this functionality. It takes sFlow packet-sample data, decodes it using Wireshark and presents it in a text format suitable for being ingested by Splunk. Install this script into the Guest VM instance where Splunk runs with:

```
[root@localhost ~]# sudo wget 'https://eos.aristanetworks.com/wiki/index.php?action=raw&title=
Configuration:Monitoring:arista_sflow_decoder' -O /opt/splunk/bin/scripts/arista_sflow
Resolving eos.aristanetworks.com... 50.19.101.37
Connecting to eos.aristanetworks.com|50.19.101.37|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1760 (1.7K) [text/x-wiki]
Saving to: `/opt/splunk/bin/scripts/arista_sflow'
```

As this script makes use of the open-source Wireshark network protocol for the packet decoding. Install that in Fedora using yum:

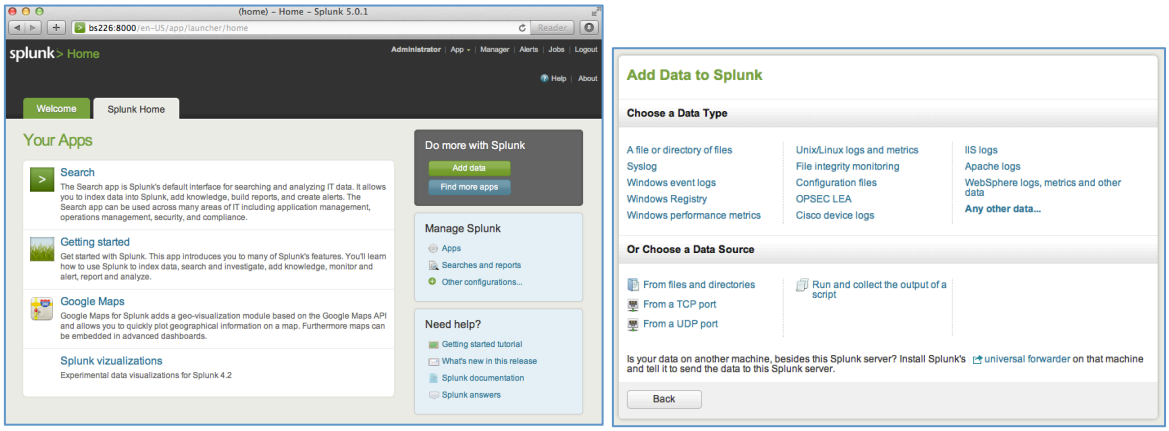
```
[root@localhost ~]# yum –y install wireshark*
```

Once sFlow is enabled on the switch you should see the sampled packets arriving in the Guest VM. You can validate this by running the **/opt/splunk/bin/scripts/arista_sflow** helper script and validating that sampled packets are appearing:

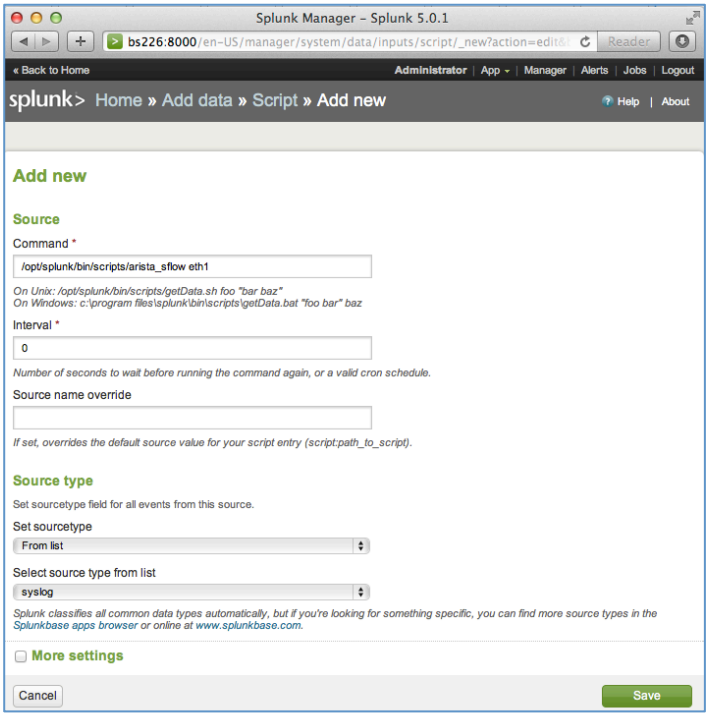
```
[root@localhost ~]# /opt/splunk/bin/scripts/arista_sflow eth1
Running as user "root" and group "root". This could be dangerous.
Capturing on eth1

t=1357542076 sflow_245.vlan.in=1 sflow_245.pri.in=0 sflow_245.vlan.out=1 sflow_245.pri.out=0 eth.dst="00:12:bf:1a:59:b2"
eth.src="b8:ff:61:c0:74:59" eth.type="0x0800" ip.version=4 ip.hdr_len=20 ip.dsfield=0 ip.dsfield.dscp="0x00" ip.dsfield.ect=0
ip.dsfield.ce=0 ip.len=64 ip.id="0x43c5" ip.flags="0x02" ip.flags.rb=0 ip.flags.df=1 ip.flags.mf=0 ip.frag_offset=0 ip.ttl=64
ip.proto=6 ip.checksum_good=1 ip.checksum_bad=0 […]
```

The final step is to enable this helper script as a data source within Splunk. In the Splunk management interface, select 'Add data' (green button below) then 'Run and collect the output of a script':



Configure the new script to call **/opt/splunk/bin/scripts/arista_sflow** as a script data source on the '**eth1**' interface, an interval of '0' and a data format of 'syslog':



At this point, Splunk is operational and data is being collected.

Summary

The extensibility of Arista EOS combined with the reporting and visualization capabilities of Splunk Enterprise provides a distributed, powerful and cost effective way to gain visibility into network traffic, with ad-hoc queries, graphs and reports capable of being generated in real time. This solution provides powerful network analysis without the need for costly hardware upgrades or proprietary network analysis modules.