Global Information Assurance Certification Paper

Full Packet Capture Infrastructure Based on Docker Containers.

GIAC (GSEC) Gold Certification

Author: Mauricio Espinosa Gomez, wiref4lcon@gmail.com
Advisor: Adam Kliarsky
Accepted: April 30th, 2016

Abstract

Full packet capture systems have become an important piece of any Organization's security infrastructure; having an exact picture of events that happened in the past is fundamental for authorized stakeholders who need to identify the cause-effect of relevant incidents. Particularly in IT security, every piece of information that flows through the network is considered a potential risk to the organization. There is no silver bullet to detect or prevent 100% of threats. Attackers are improving their methodologies to circumvent protective technologies with sophisticated evasion techniques. In most cases when an incident has already occurred, a Full Packet Capture (FPC) not only provides the history of the events in question, but might also provide interesting correlations with other phases of the incident that aren't easy to identify at first glance. One of the biggest challenges in today's Organizations is the cost of having an effective Centralized Full Packet Capture Infrastructure (CFPCI). Commercial solutions are quite expensive and can be hard to implement, here is where a good combination of Open Source Technologies such as Moloch, Puppet and Docker can help to minimize cost and effectively fill the gap.

1. Introduction

In today's world, it is common to hear news about organizations being breached by malicious actors, even in highly protected environments; the risk of being exploited is always present, when an incident has already occurred, a full packet capture provides invaluable information to effectively backtrack the event in question. "Full packet capture data provides a full accounting for every data packet transmitted between two endpoints. "If we compare the investigation of computer-related crime to human-focused crime, having FPC data from a device under investigation would be equivalent to having a surveillance-video recording of a human suspect under investigation. Ultimately, if an attacker accesses a system from the network, there will be evidence of it within FPC data" (Sanders & Smith, 2013).

"FPC can be overwhelming due to its completeness, but it is this high degree of granularity that is valuable when it comes to providing analytic context" (Smith, Sanders, 2013), this context is not only useful for incident responders, but also works as an important and often delicate evidence for involved units in the organization, the most common are: IT Security, IT Operations, Network Operations, Human Resources and Legal[1].

The most common form of FPC data is the PCAP data format, it is supported by most open source solutions and has been the gold standard for FPC data for quite a while. The most popular library that allows applications to interact with network interfaces is Libpcap (Sanders & Smith, 2013).

There will be costs involved when planning a FPC infrastructure, depending on the total throughput and retention requirements, organizations should consider rightsizing storage, RAM, CPU and Network interfaces, the most common factor when generating FPC data is storage. PCAP files takes up a lot of space relative to all other data types; so determining the amount of storage is one of the most important aspects of the initial planning (Sanders & Smith, 2013).

[1] Prior to implementing a FPC infrastructure, it is recommended to consult the project scope with Legal and the local Country regulations.

There are a variety of commercial solutions that offer unique features for FPC, unfortunately they tend to be very expensive, sometimes hard to implement and sometimes inflexible. The benefit, nonetheless; is their guaranteed long term support. On the other hand, there are open source FPC solutions, the most important benefit is low cost, software is free and unlimited, world wide organizations may save up to 6 numbers when implementing Open Source Vs Commercial FPC solutions, other important feature is flexibility, it is possible to customize the technology to suit specific needs for different computing environments, or even combine it with other open source technologies to create new solutions. The big challenge, however; is the right headcount to support the tool and make it work.

There are other important factors to consider when implementing FPC, virtualization technologies have become very popular in Datacenters, the benefit of having private, hybrid or public Clouds, is also raising the bar of complexity for most organizations. Your FPC strategy might fall into different computing environments with different needs, here is where adaptability comes in to play. Automated provisioning and configuration management can be a difficult task to accomplish if we have VMs and Barebones in different Geographic locations. A well planned FPC approach should solve this problem with portable applications that can be rapidly be deployed and orchestrated in different computing ecosystems.

Other important factor is security, upgrades and patching can be a nightmare if we have different applications in non-isolated environments, container technologies are quite useful for this cases, the benefit of having disposable applications that can be upgraded without touching inner components of the host system provides an efficient way to automate upgrades and patching without downtimes.

In general, there are two main approaches for implementing a full packet capture infrastructure: decentralized and centralized. The most common is decentralized and involves placing independent capturers in the network, if an event occurs, the analyst should have to login to each standalone capturer to get the specific network traffic information, this architecture works well for small networks where traffic segregation is limited to 10 FPC points. The centralized architecture consists of different aggregators or capturers that connect to a central data-aggregation instance and provides a single point of visibility for the

entire network traffic, this scales well for medium-large networks and provides an effective way for security analytics, one of the caveats, however, is the single point of failure, in most cases if the data-aggregator goes down, the network visibility will be lost, there are ways to circumvent this scenario using High Availability (HA) related technologies, such as: clustering, load balancing, failover, etc.

2. Overview of the tools

2.1 Moloch

Among FPC open source solutions, Moloch stands as one of the few that provides a complete FPC Framework, Moloch was developed by Andy Wick and Eoin Miller both members of AOL's emergency response team. "Andy Wick has more than 15 years of development experience at AOL. He has recently come into the CERT group and has begun developing tools for defense and forensics. Eoin Miller specializes in using IDS and full packet capture systems to identify drive by exploit kits and the traffic that feeds them. He regularly contributes to the development of signatures for Emerging Threats/OISF" (Shmoocon 2013 - Moloch A New And Free Way To Index Your Packet Capture, 2013).

Moloch is capable of handling high speed Networks and its very flexible for specific needs, some of its main features are: scalable IPv4 packet capturing (PCAP), Indexing and Database System powered by Elastic Search, and a wonderful WEB GUI. (Moloch, n.d.)Moloch includes three main components: the "Capturer" which sniffs the network interface, parses the traffic and creates the Session Profile Information (SPI) data, the "Elastic Search Database" used to store and search through the SPI data; and the "Viewer" who provides the web interface that allows GUI and API access from remote hosts to browse or query SPI data and retrieve stored PCAP files. Moloch can be adapted for both centralized and decentralized architectures. For the purpose of this paper we will use the centralized or "Multiple Hosts Monitoring Multiple Segments" architecture (Moloch Architecture, 2014). This involves multiple capturers placed in different network segments, each one receiving traffic from "Mirrored" or "Tapped" ports and aggregating data to one central location.

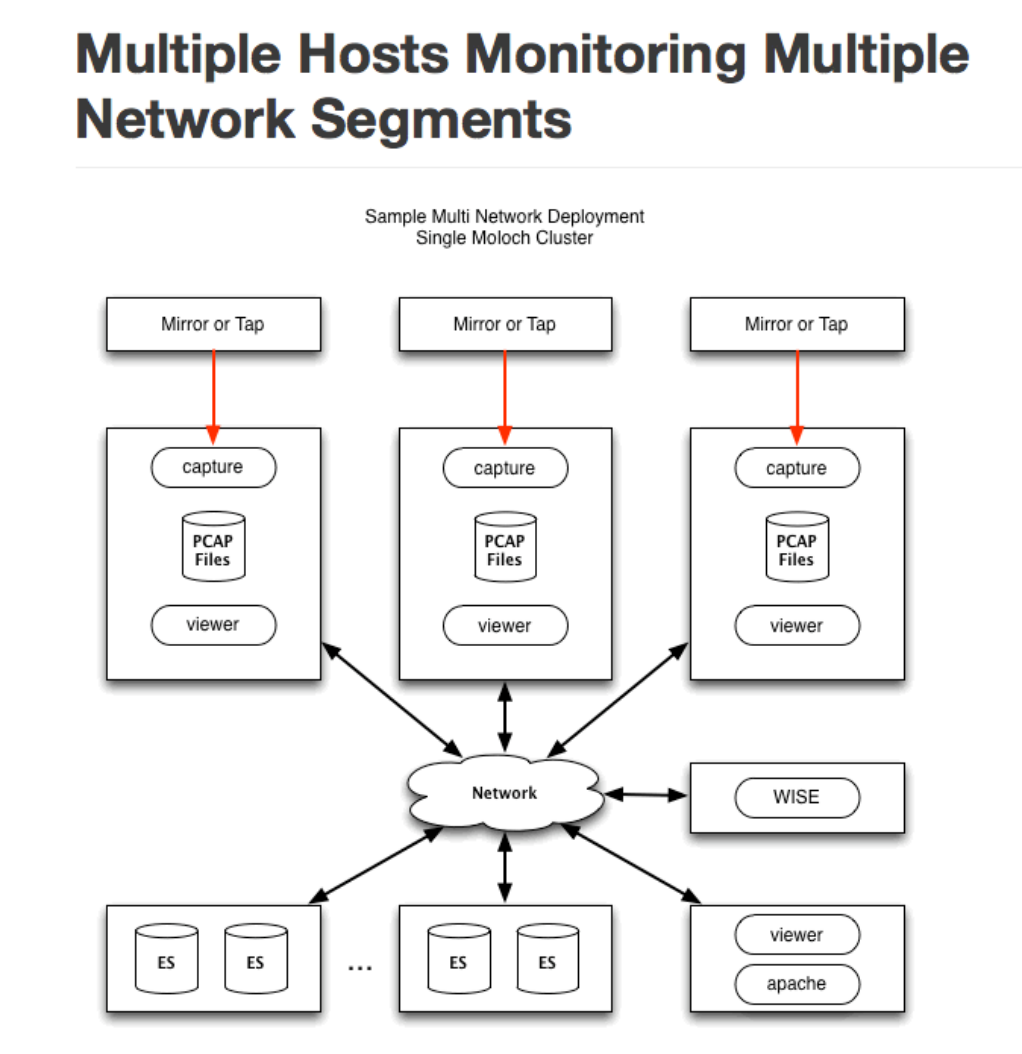


Figure 1 – Sample Multi Network Deployment Single Moloch Cluster from https://github.com/aol/moloch/wiki/Architecture

2.2 Docker

Developed by Solomon Hykes, and released in March 2013 under the Apache 2.0 license, Docker stands as an open standard platform for developing, packaging and running portable distributed applications. With Docker, it is possible to build, ship and run applications on any platform such as PCs, clouds, data centers or virtual machines. Docker provides a simplistic way for packaging all the required applications with their respective

dependencies, libraries and system tools, into one piece of software unit, called Docker image (Vohra, 2015).

Docker images are different from virtual machine images, the main benefit is resource optimization, while virtual machines run on a separate guest OS, Docker images run within the same OS kernel, this means; multiple isolated environments called “Docker containers” can be running simultaneously in the same host sharing the same kernel (Vohra, 2015).

Some of the benefits of using containers are: encapsulation, portability, reduced overhead and scalability. Encapsulation means everything needed by the application can be packaged within the container, this includes dependencies and environment variables. Containers are Linux “Distro” and platform agnostic, which makes them portable to any environment. They also provide reduced overhead associated with server density (containers are typically $1/10^{th}$ to $1/100^{th}$ the size of the equivalent application packaged within a VM). Scalability means containers can be dynamically reduced or expanded, this can be applied to either one or multiple instances through centralized orchestration.

Docker can be useful with high performance applications (HPA), FPC is a good example of HPA. Consider three Moloch capturers in separate containers running in the same host, each container sniffing in separate 1Gbs network interfaces, on peak hours the host might experience a substantial resource contention for Memory, CPUs and block I/O, if the host is a VM on top of a hypervisor; the hypervisor might also experience unexpected resource contention and perhaps hit the performance of other VMs within the same virtual ecosystem. We know there are explicit OS native methods to address this issue (local cgroups), with Docker, is fairly easy to ensure Quality of Service (QoS) for HPA without affecting external entities beyond the containment space.

In addition, consider installing a honeypot and IDS in the FPC host, this is often useful to deter, detect or prevent an attacker of gaining unauthorized access to critical forensic data, most honeypots are limited to run “out of the box” in Debian/Ubuntu, if our internal policies dictate RHEL/Centos, you might dedicate a considerable amount of time to make the application work in Red Hat,

Docker can easily address this issue leveraging a flexible space for running packages compiled under different Linux “Distros“ in the same host.

2.3 Puppet

Developed by Luke Kanies, founder of Puppet Labs in 2005. Puppet stands as a powerful configuration management system, Puppet can handle hundreds of thousands of nodes from central or distributed locations, Puppet manages data, including users, packages, processes, services and any imaginable component of the system. It is very flexible and robust, capable of managing complex and distributed components to ensure consistency and availability across different OS platforms (Rhett, 2013).

Some of its main features are: “Deployment of new systems with consistent configuration and data installed, upgrade security and software packages across the enterprise, roll out new features and capabilities to existing systems painlessly, adjust system configurations to make new data sources available, decrease the cost and effort involved in minor changes on hundreds of systems, simplify the effort and personnel involved in software deployments, automate build-up and tear-down of replica systems to test proposed changes, repurpose existing computer resources for a new use within minutes, gather a rich data set of information about the infrastructure and computing resources, provide a clear and reviewable change control mechanism” (Rhett, 2013)

For this demonstration, we will show how Puppet can be used to build and enforce the configuration of Docker containers; this method can be used in either: “Agent/Master” or “Stand-Alone” architectures accordingly. (Overview of Puppet’s Architecture, 2016).

3. Lab setup

For the purpose of this paper, a simple FPC configuration model will be shown, we will use a small virtual environment running on a laptop simulating the core components of a centralized FPC infrastructure, this model, however; can be scaled to hundreds of nodes, and can be quite efficient if we compare it with commercial solutions.

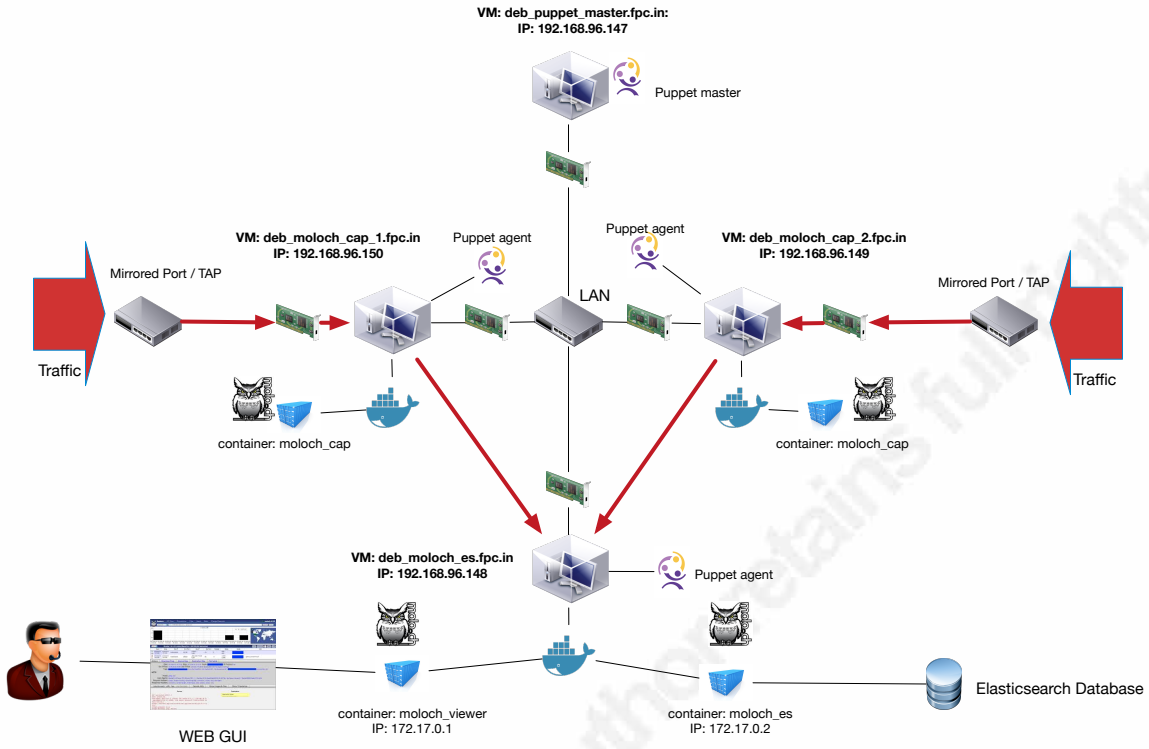


Figure 2 – Lab Setup

Lab components:

VMware fusion: 8.1.0

1 Virtual machine acting as the Puppet Master server (Configuration manager):

- OS: Debian 8.3.0
- Hostname: deb_puppet_master.fpc.in
- CPU: 1
- RAM: 2048 MB
- HD: 20 GB
- IP: 192.168.96.147
- Applications: puppetmaster-passenger (3.7.2)

1 Virtual machine acting as the Elastic Search Database and Moloch's viewer (GUI):

- Hostname: deb_moloch_es.fpc.in
- OS: Debian 8.3.0
- CPU: 1
- RAM: 2048 MB
- HD: 20 GB
- NIC: 1

IP: 192.168.96.148
Applications: puppet agent (3.7.2)

2 Virtual machines acting as Moloch Capturers (aggregators):

Hostname 1: deb_moloch_cap_1.fpc.in
Hostname 2: deb_moloch_cap_2.fpc.in
OS: Debian 8.3.0
CPU: 1
RAM: 1024 MB
HD: 20 GB
NIC 1: IP: 192.168.96.150
NIC 2: SPAN / Mirror
NIC 1: IP: 192.168.96.149
NIC 2: SPAN / Mirror
Applications: puppet agent (3.7.2)

Note: we won't go into details of how to install a Debian 8 VM in VMware Fusion, at this point, you should have your 4 VM's up and running in the same VLAN (Virtual LAN) with Internet connection accordingly. Ensure the capturers have the second NIC up and running in promiscuous mode.

Your hypervisor setup should look similar to the following image:

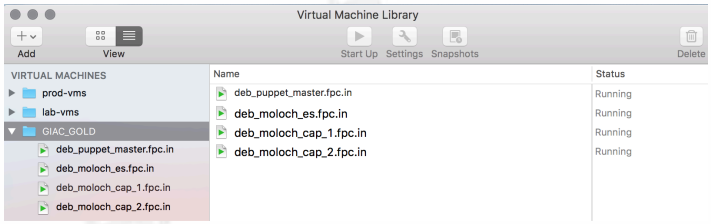


Figure 3 – Virtual Machine Layout

Note: All the following commands should be executed as root.

Puppet Master setup:

1. Update and upgrade packages:

```
apt-get update && apt-get upgrade
```

2. Set timezone to UTC:

```
dpkg-reconfigure tzdata
```

```
apt-get install –y ntpdate && ntpdate pool.ntp.org
```

3. Install useful packages:

```
apt-get -y install vim curl git tree mc
```

4. Assign FQDN to "/etc/hostname" and "/etc/hosts" files:

```
root@deb_puppet_master:~# cat /etc/hosts
127.0.0.1       localhost
192.168.96.147  deb_puppet_master.fpc.in        deb_puppet_master       puppet

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
root@deb_puppet_master:~# _
```

Figure 4 – Sample /etc/hosts File

5. Reboot for changes to take effect.
6. Install the Puppet master server and agent, for this demonstration, we will use Puppet's Passenger version (Configuring a Puppet Master Server with Passenger and Apache, n.d.)which is scalable to hundreds of nodes in production environments:

```
apt-get install puppetmaster-passenger
```

7. For this demonstration we will use version 3.7.2:

```
root@deb_puppet_master:~# puppet --version
3.7.2
```

Figure 5 – Getting the Puppet Version

8. Start the Puppet Master service:

```
puppet master
```

9. Install the "garethr/docker" module developed by Gareth Rushgrove for managing Docker from Puppet: (garethr/docker, n.d.)

```
puppet module install garethr-docker
```

10. After installation, the "docker" module should appear in Puppet's "modules" folder:

```
 Left     File     Command     Options     Right
<- /etc/puppet/modules                    .[^]>
 .n          Name          Size    Modify time
 /..                     UP--DIR  Mar 21 03:26
 /apt                       4096  Feb 29 21:43
 /docker                    4096  Mar 21 04:30
 /epel                      4096  Dec  5 16:42
 /stdlib                    4096  Jan 12 11:08
```

Figure 6 – Puppet Modules

11. Create an additional folder where Moloch files will be hosted:

```
mkdir -p /etc/puppet/modules/docker/files
```

12. The directory structure with the necessary files for this demonstration is publicly available on Github, run the following commands to clone it to your local instance:

```
cd /etc/puppet/
git clone https://github.com/wiref4lcon/giac_moloch.git
cp –fr giac_gold/* .
```

Note: Your directory structure inside "/etc/puppet/" should look like the following image:

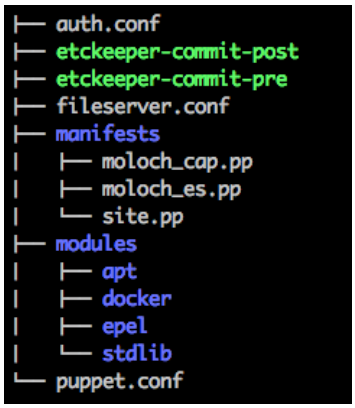


Figure 7 – Puppet Directory Structure

Note: The "docker" module will be enforcing two different profiles: "moloch_es" (Elasticsearch Database) and "moloch_cap" (Capturer), each one will have different configuration files. The "moloch_cap" and "moloch_es" folders will hold the necessary components for Docker to build images using the "Dockerfile". The "cap_etc" and "es_etc" folders will hold the configuration files for each application accordingly, if any of this files are modified (the md5 checksum changes), Puppet will recreate the container with the new changes.

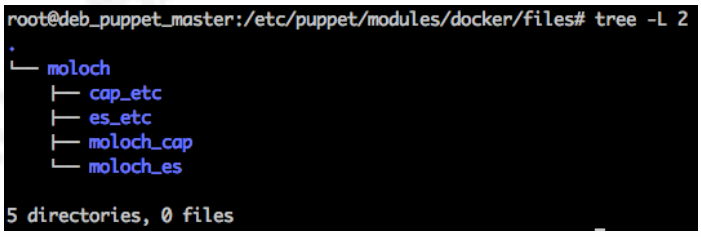


Figure 8 – Reflected Changes

Connect "deb_moloch_es.fpc.in" to the Puppet Master server:

Once we have the Puppet Master ready to start enforcing configurations, we can proceed to install and connect the Puppet agents of the three VMs that will be part of the centralized full packet capture model, the following process should be performed separately in each VM; for this demonstration, we will show how to connect the VM “deb_moloch_es” to our Puppet Master server:

In the VM named “deb_moloch_es.fpc.in”, perform the following process:

1. Assign FQDN of the hostname in “/etc/hostname” and include the following entries in “/etc/hosts”:

```
root@deb_moloch_es:/home/vmware# cat /etc/hosts
127.0.0.1       localhost
192.168.96.148  deb_moloch_es.fpc.in
192.168.96.147  puppet
192.168.96.149  deb_moloch_cap_2.fpc.in
192.168.96.150  deb_moloch_cap_1.fpc.in
```

Figure 9 – Sample /etc/hosts File

2. Set timezone to UTC:

```
dpkg-reconfigure tzdata
apt-get install ntpdate && ntpdate pool.ntp.org
```

3. Reboot to apply changes:
4. Update, upgrade and install the Puppet agent:

```
root@debian:/home/vmware# apt-get update && apt-get upgrade
root@debian:/home/vmware# apt-get install puppet
```

5. Ensure no previous certificates are in place:

```
root@deb_moloch_es:/home/vmware# rm –rf /var/lib/puppet
```

6. Send the certificate request to the Puppetmaster (acting as the CA):

```
root@deb_moloch_es:/home/vmware# puppet agent -t
```

7. In the VM “deb_puppet_master.fpc.in” ensure no previous related certificate is in place and accept the new request by signing the certificate:

```
root@deb_moloch_puppet_master:~/# puppet cert clean deb_moloch_es.fpc.in &&
puppet cert sign deb_moloch_es.fpc.in
```

```
root@deb_puppet_master:/home/vmware# puppet cert clean deb_moloch_es.fpc.in
Error: Could not find a serial number for deb_moloch_es.fpc.in
root@deb_puppet_master:/home/vmware# puppet cert sign deb_moloch_es.fpc.in
Notice: Signed certificate request for deb_moloch_es.fpc.in
Notice: Removing file Puppet::SSL::CertificateRequest deb_moloch_es.fpc.in at '/var/lib/puppet/ssl/ca/
requests/deb_moloch_es.fpc.in.pem'
root@deb_puppet_master:/home/vmware#
```

Figure 10 – Sign the Certificate

8. In the "deb_moloch_es.fpc.in" VM run the following command to start the system configuration enforcing process:

```
puppet agent -t
```

Note: if you get the following error:

```
Error: Could not request certificate: The certificate retrieved from the master does not match the agent's private key.
Certificate fingerprint: E5:A4:70:1B:EC:2B:B8:32:5E:83:13:93:F4:AA:B9:E1:68:24:3E:25:1E:A9:C8:B7:DC:8E:F9:EC:C9:6F:A7:4C
To fix this, remove the certificate from both the master and the agent and then start a puppet run, which will automatically
te.
On the master:
  puppet cert clean deb_moloch_es.fpc.in
On the agent:
  1a. On most platforms: find /var/lib/puppet/ssl -name deb_moloch_es.fpc.in.pem -delete
  1b. On Windows: del "/var/lib/puppet/ssl/deb_moloch_es.fpc.in.pem" /f
  2. puppet agent -t
```

Figure 11 – Sample Error

Make sure the time matches between both hosts and run the following commands:

In deb_puppet_master.fpc.in:

```
puppet cert clean deb_moloch_es.fpc.in
```

In deb_moloch_es.fpc.in:

```
rm –rf /var/lib/puppet && puppet agent –t
```

9. Once completed, the "moloch-es" and "moloch-viewer" containers should be up and running:

```
root@deb_moloch_es:/home/vmware# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
                                        NAMES
ebbab016eaa6        moloch_es           "/usr/bin/supervisord"   4 minutes ago       Up 4 minutes        0.0.0.0:
9200->9200/tcp, 0.0.0.0:9300->9300/tcp   moloch-es
80e16b6f1450        moloch_viewer       "/usr/bin/supervisord"   4 minutes ago       Up 4 minutes
                                        moloch-viewer
```

Figure 12 – Verify Containers

10. Verify the container's IP with the following command:

```
docker exec moloch-es ifconfig
```

Note: If the IP isn't "172.17.0.2" you will need to replace the three entries in the following location inside the Puppet Master server:

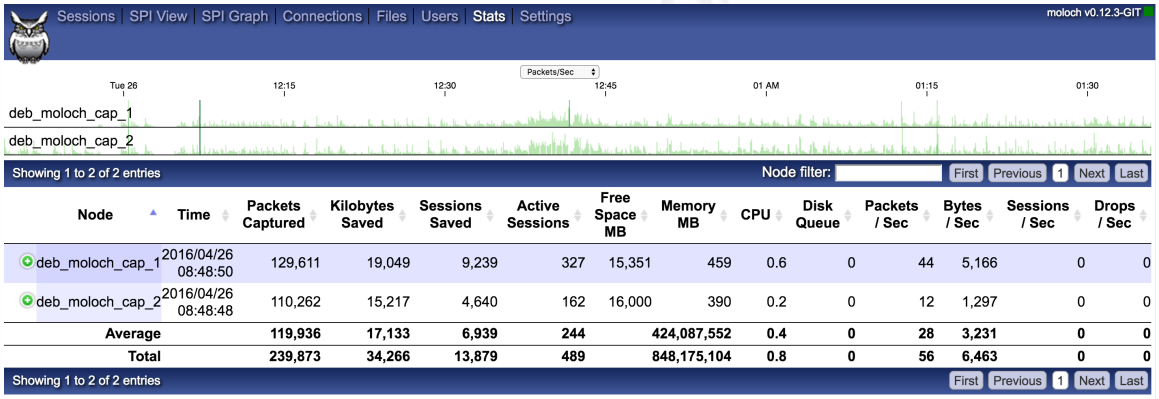
"/etc/puppet/modules/docker/files/es_etc/elasticsearch.yml" and restart both containers with the following commands:

```
docker restart moloch-es && docker restart moloch-viewer
```

Note: Repeat steps 1 to 8 on the virtual machines: "deb_moloch_cap_1.fpc.in" and "deb_moloch_cap_2.fpc.in" (make sure you have the second NIC of each capturer are up before step 8), once completed, you should have a centralized full packet capture infrastructure based on Docker containers.

Open your browser (Preferably Firefox) and go to the following URL: https://192.168.96.148:8005/stats
Enter username: admin and password: admin

The following screen shows Moloch's "Stats" section, where two capturers are sending metadata to the central instance:

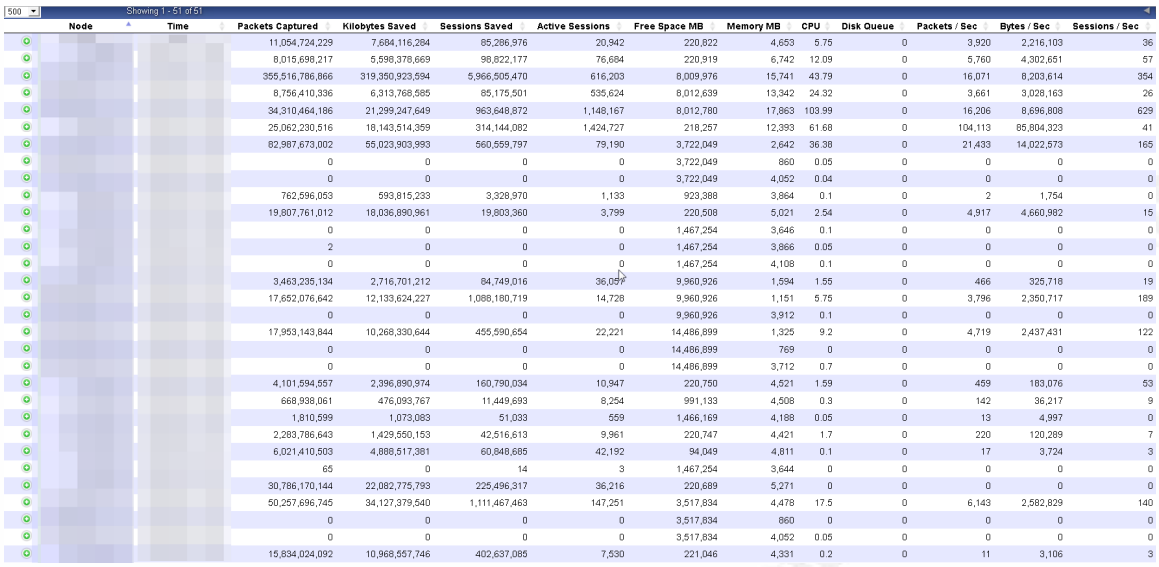


Node	Time	Packets Captured	Kilobytes Saved	Sessions Saved	Active Sessions	Free Space MB	Memory MB	CPU	Disk Queue	Packets / Sec	Bytes / Sec	Sessions / Sec	Drops / Sec
deb_moloch_cap_1	2016/04/26 08:48:50	129,611	19,049	9,239	327	15,351	459	0.6	0	44	5,166	0	0
deb_moloch_cap_2	2016/04/26 08:48:48	110,262	15,217	4,640	162	16,000	390	0.2	0	12	1,297	0	0
Average		**119,936**	**17,133**	**6,939**	**244**	**424,087,552**		**0.4**	**0**	**28**	**3,231**	**0**	**0**
Total		**239,873**	**34,266**	**13,879**	**489**	**848,175,104**		**0.8**	**0**	**56**	**6,463**	**0**	**0**

Showing 1 to 2 of 2 entries First Previous 1 Next Last

Figure 13 – Moloch Stats

This model can scale to hundreds of nodes aggregating metadata to a central location:



Node	Time	Packets Captured	Kilobytes Saved	Sessions Saved	Active Sessions	Free Space MB	Memory MB	CPU	Disk Queue	Packets / Sec	Bytes / Sec	Sessions / Sec
		11,054,724,229	7,684,116,284	85,286,976	20,942	220,822	4,653	5.75	0	3,920	2,216,103	36
		8,015,698,217	5,598,378,669	98,822,177	76,684	220,919	6,742	12.09	0	5,760	4,302,651	57
		355,516,786,866	319,350,923,594	5,966,505,470	616,203	8,009,976	15,741	43.79	0	16,071	8,203,614	354
		8,756,410,336	6,313,768,585	85,175,501	535,624	8,012,639	13,342	24.32	0	3,661	3,028,163	26
		34,310,464,186	21,299,247,649	963,648,872	1,148,167	8,012,780	17,863	103.99	0	16,206	8,696,808	629
		25,062,230,516	18,143,514,359	314,144,082	1,424,727	218,257	12,393	61.68	0	104,113	85,804,323	41
		82,987,673,002	55,023,903,993	560,559,797	79,190	3,722,049	2,642	36.38	0	21,433	14,022,573	165
		0	0	0	0	3,722,049	860	0.05	0	0	0	0
		0	0	0	0	3,722,049	4,052	0.04	0	0	0	0
		762,596,053	593,815,233	3,328,970	1,133	923,388	3,864	0.1	0	2	1,754	0
		19,807,761,012	18,036,890,961	19,803,360	3,799	220,508	5,021	2.54	0	4,917	4,660,982	15
		0	0	0	0	1,467,254	3,646	0.1	0	0	0	0
		2	0	0	0	1,467,254	3,866	0.05	0	0	0	0
		0	0	0	0	1,467,254	4,108	0.1	0	0	0	0
		3,463,235,134	2,716,701,212	84,749,016	36,097	9,960,926	1,594	1.55	0	466	325,718	19
		17,652,076,642	12,133,624,227	1,088,180,719	14,728	9,960,926	1,151	5.75	0	3,796	2,350,717	189
		0	0	0	0	9,960,926	3,912	0.1	0	0	0	0
		17,953,143,844	10,268,330,644	455,590,654	22,221	14,486,899	1,325	9.2	0	4,719	2,437,431	122
		0	0	0	0	14,486,899	769	0	0	0	0	0
		0	0	0	0	14,486,899	3,712	0.7	0	0	0	0
		4,101,594,557	2,396,890,974	160,790,034	10,947	220,750	4,521	1.59	0	459	183,076	53
		668,938,061	476,093,767	11,449,693	8,254	991,133	4,508	0.3	0	142	36,217	9
		1,810,599	1,073,083	51,033	559	1,466,169	4,188	0.05	0	13	4,997	0
		2,283,786,643	1,429,550,153	42,516,613	9,961	220,747	4,421	1.7	0	220	120,289	7
		6,021,410,503	4,888,517,381	60,848,685	42,192	94,049	4,811	0.1	0	17	3,724	3
		65	0	14	3	1,467,254	3,644	0	0	0	0	0
		30,786,170,144	22,082,775,793	225,496,317	36,216	220,689	5,271	0	0	0	0	0
		50,257,696,745	34,127,379,540	1,111,467,463	147,251	3,517,834	4,478	17.5	0	6,143	2,582,829	140
		0	0	0	0	3,517,834	860	0	0	0	0	0
		0	0	0	0	3,517,834	4,052	0.05	0	0	0	0
		15,834,024,092	10,968,557,746	402,637,085	7,530	221,046	4,331	0.2	0	11	3,106	3

Figure 14 – Scaled Deployment

Moloch offers a powerful interface to query metadata using BPF's (Berkeley Packet Filters). It is also possible to browse payloads content and export them to PCAP files.

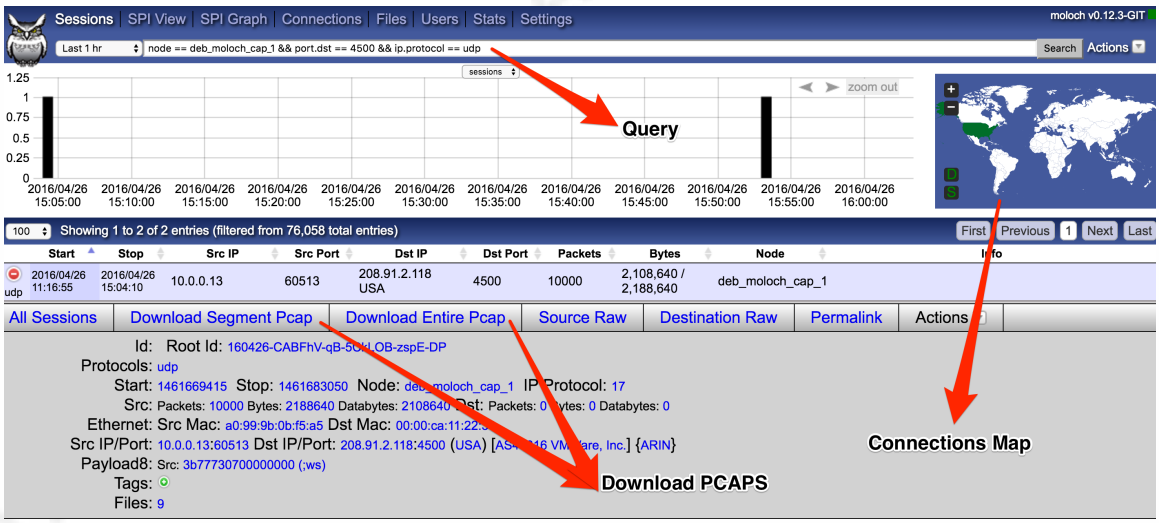


Figure 15 – Moloch Interface

It is important to note PCAPS are stored locally in each aggregator (Moloch capturer), when the "Download Segment Pcap" or "Download Entire Pcap" options are selected, the "Moloch Viewer" process of the centralized instance will request the remote "Moloch Viewer" to handle directly to the user the PCAP file in question. For the purpose of this demonstration we used the

"Bad fix" cited in the "Viewer issues with multiple nodes" (Wick, 2016), for production environments, a PKI infrastructure is recommended.

4. Conclusion:

The model presented in this paper demonstrates how effective open source solutions can be to proactively leverage a cost-effective; centralized full packet capture infrastructure. As one of a kind; Moloch offers a simplistic, but powerful forensic approach for network surveillance, capable of processing hundreds of billions of records, and the ability to run queries on a nice WEB GUI. Automation is a fundamental piece for operational resiliency; even with a Disaster Recovery Plan (DRP), unexpected downtimes can have an impact over the initial stages of an Incident Response Plan (IR), Puppet can help to mitigate this risk, with a flexible configuration enforcing framework, reacting to events and keeping the desired state of services. The inclusion of Docker containers to the Linux world has opened a wide horizon of emerging technologies, the biggest win of containers is: "Minimalistic Isolation", which increases the portability and orchestration of distributed applications to optimized cloud environments, particularly for the exposed centralized full packet capture infrastructure model, containers can provide resource contention to the network level and disposable aggregators that can be recreated or upgraded in seconds.

3. References

Configuring a Puppet Master Server with Passenger and Apache. (n.d.). Retrieved from puppet.com: https://docs.puppetlabs.com/guides/passenger.html

Dusia, A., Yang, Y., & Taufer, M. (2015). Network Quality of Service in Docker Containers. Newark, DE: IEEE. doi:10.1109/CLUSTER.2015.96

garethr/docker. (n.d.). Retrieved from forge.puppetlabs.com: https://forge.puppetlabs.com/garethr/docker

McDaniel, S., Herbein, S., & Taufer, M. (2015). A Two-Tiered Approach to I/O Quality of Service in Docker Containers. (p. 2). Newark, DE: IEEE. doi:10.1109/CLUSTER.2015.96

Moloch. (n.d.). Retrieved from Moloch: http://molo.ch

Moloch Architecture. (2014). Retrieved from Github: https://github.com/aol/moloch/wiki/Architecture

Overview of Puppet's Architecture. (2016). Retrieved from puppet.com: https://docs.puppet.com/puppet/4.4/reference/architecture.html

Rhett, J. (2013). *Instant Puppet 2 Starter.* Birmingham B3 2PB, UK.: Packt Publishing.

Sanders, C., & Smith, J. (2013). *Applied Network Security Monitoring.* Elsevier Science, Syngress.

Shmoocon 2013 - Moloch A New And Free Way To Index Your Packet Capture . (2013). Retrieved from http://www.securitytube.net: http://www.securitytube.net/video/7187

Vohra, D. (2015). *Pro Docker.* Appress.

Wick, A. (2016, 04 05). *FAQ*. Retrieved from Github.com: https://github.com/aol/moloch/wiki/FAQ

Xavier, M., Neves, M., Rossi, F., Ferreto, T., Lange, T., & De Rose, C. (2013). Performance Evaluation of Container-based Virtualization for High Performance Computing Environments., *21*, p. 8. Porto Alegre, Brazil. doi:10.1109/PDP.2013.41