Interested in learning more about security?

SANS Institute InfoSec Reading Room

Building a Home Network Configured to Collect Artifacts for Supporting Network Forensic Incident Response

A commonly accepted Incident Response process includes six phases: Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned. Preparation is key. It sets the foundation for a successful incident response. The incident responder does not want to be trying to figure out where to collect the information necessary to quickly assess the situation and to respond appropriately to the incident. Nor does the incident responder want to hope that the information he needs is available at the level of det...

Building a Home Network Configured to Collect Artifacts for Supporting Network Forensic Incident Response

GIAC (GCIA) Gold Certification

Author: Gordon Fraser, Gordon.fraser@ctipc.com
Advisor: Richard Carbone

Accepted: September 19, 2016

Abstract

A commonly accepted Incident Response process includes six phases: Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned. Preparation is key. It sets the foundation for a successful incident response. The incident responder does not want to be trying to figure out where to collect the information necessary to quickly assess the situation and to respond appropriately to the incident. Nor does the incident responder want to hope that the information he needs is available at the level of detail necessary to most effectively analyze the situation so he can make informed decisions on the best course of action. This paper identifies artifacts that are important to support network forensics during incident response and discusses an architecture and implementation for a home lab to support the collection of them. It then validates the architecture using an incident scenario.

1. Introduction

A commonly accepted Incident Response (IR) process includes six phases: Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned (Skoudis, Strand, and SANS, 2014). Preparation is key. It sets the foundation for a successful incident response. The incident responder does not want to be trying to figure out where to collect the information necessary to quickly assess the situation and to respond appropriately. Nor does the incident responder want to hope that the information he needs is available at the level of detail necessary to most effectively analyze the situation so he can make informed decisions on the best course of action.

An important component of the IR Preparation Phase is to determine the types of information that are potentially valuable in an incident response scenario. What log and configuration files should be collected? What information should be captured in the log files? How long should log files be retained? What are the organization's data retention policies? What configuration management process should be followed when configuration files are changed? This paper identifies key artifacts that are important to support network forensics during incident response. It discusses the setup of a home lab architected to collect the artifacts using open source tools and validates the implementation through a test scenario.

The potential impact of not including these artifacts when preparing for an incident is to assume the risk that critical information is not available when it is needed. Even if the data exists, there is the risk that the incident responder is slowed down while he is trying to figure out where to get the information and how to collect it. These are risks that an organization can ill afford to take. The organization should make adequate preparations to avoid and mitigate these risks. After all, the probability that the organization will be compromised at some point in time is high. When it is, these artifacts may be important in determining how to respond appropriately.

In addition to identifying the artifacts, consideration needs to be made on how long the data is retained by the organization and how configuration management is done with respect to configuration files. According to Mandiant in their M-Trends Report, the

median time it took for an organization to discover a breach or hear about it from an external organization was 146 days from the time of compromise (Mandiant, 2016). The median number of days was 56 days if the organization discovered the breach themselves. When notification came from an external source the median number of days until discovery jumped to 320 days. These figures give an indication of the period of time the artifacts need to be retained to be useful for the investigation.

This paper focuses on the artifacts to collect and not the configuration management processes or data retention policies of the organization. When configuration files are included in the artifacts, the assumption is that the configuration files corresponded to the time period when the network traffic data was captured.

2. Network Artifacts

Artifacts which provide evidence or insight into network communications can be found in many places. Dynamic Host Configuration Protocol (DHCP) servers, Domain Name System (DNS) servers, Web Proxy Servers, Intrusion Detection Systems (IDS), and firewalls all can generate artifacts which can be helpful when responding to an incident. The key here is that systems must be configured to generate and capture artifacts and must be available to the incident responder when needed. This section examines common artifact sources to consider when developing an incident response plan during the IR Preparation Phase. This paper focuses on examples from Linux. Equivalent or similar artifacts are available for other Operating Systems.

In addition to the artifacts, the configuration files themselves should be collected so that the parameters concerning their collection are understood. For example, when looking at DHCP logs, the analyst would want to know the duration of the lease so that he would know when to expect lease renewal transactions to take place. The absence of a lease renewal would indicate the computer left the network.

2.1. Network Time Protocol (NTP)

Time should be synchronized between different systems to allow for the effective correlation of information generated from them. Synchronizing time saves the incident

responder much frustration and work by not requiring him to try and correlate times between artifacts generated from different systems. The Network Time Protocol (NTP) was developed to provide accurate time services on the network and to allow for consistency among computers on a network.

2.2. Dynamic Host Configuration Protocol (DHCP)

For those computers that do not have a statically assigned IP address, DHCP provides a computer an IP address and other network configuration information. The computer must contact the DHCP server to be assigned an IP address before it can send data on the network. Because DHCP traffic is predictable, DHCP logs can be an excellent source of information during incident response. From these logs, the analyst can determine when a computer joined the network, was present on the network, and the time frame when it left the network.

2.3. Domain Name System (DNS)

DNS translates human-readable host names to IP addresses. It is a fundamental service of Internet communications. Prior to initiating communication to another computer based on the host name, the computer queries the DNS server to translate the host name to an IP address. By examining DNS request/response traffic, an incident responder can gain valuable information including, when communication with a particular host began since the first step in the communication process generally is to resolve the hostname to an IP address. It can be an indicator of who else might have also been compromised by virtue of the fact that they also queried DNS to resolve the same host name. In the event of multiple systems being compromised, records of DNS queries can provide a lead to the initial vector of compromise by virtue of it being the first request to resolve the host name.

Most DNS servers allow query logging, but not response logging (Hagen, 2015c). While only logging DNS queries answers some questions about the network traffic, it leaves out the responses. The absence of responses may be acceptable in some circumstances, but in others, like cache poisoning or fast flux, it omits information that is critical to understanding what is going on.

An alternative to DNS logging could be full packet capturing of DNS traffic. A disadvantage of relying on packet capturing is the preprocessing that would be required before analysis can take place (Hagen, 2015c). A better alternative would be to implement a third party tool to perform passive DNS monitoring, which captures and logs both requests and responses. PassiveDNS available from https://github.com/gamelinux/passivedns is one such tool. (Hagen, 2015c; Fjellskål, December 2015). This tool can monitor a network interface or read from a Pcap file to generate DNS log entries that include both requests and responses.

2.4. Proxy Server Logs

A proxy server brokers traffic between a client and a server and are frequently associated with web traffic. Many organizations use a proxy server between their internal network and the Internet. They capture information that can be of value to the incident responder. Proxy logs capture web traffic requests and response. They also cache copies of resources retrieved from the web servers. The proxy cache may have copies of files, like malware, that was retrieved from a web server (Hagen, 2015d). Squid is a popular open source web proxy.

2.5. Firewall Logs

Firewalls are a specialized router designed to perform packet inspection and make decisions on what traffic should be forwarded, logged, and blocked (Davidoff and Ham, 2012). Firewalls are flexible and can be configured to log traffic at various levels of detail based on the needs of the organization. All traffic that has been denied based on the firewall rules can be logged. Traffic, meeting specific conditions, can be logged. All traffic permitted through the firewall can be logged. The IR Preparation Phase should include defining what information is logged by the firewall to ensure it is available to support incident response.

Another consideration of firewall logging is that during an incident, the organization may make the decision to initially monitor an attack rather than just cut it off to gather more information to determine the course of action in responding to the incident. Firewall logging can assist with this. Special rules can be added during the

incident to log traffic associated with the incident (Davidoff and Ham, 2012). Part of the IR Preparation Phase might be to establish protocols for doing so. Iptables is a standard Linux based firewall.

2.6. Intrusion Detection System

Intrusion Detection Systems (IDS) are a common component of many corporate environments. An IDS examines network traffic that crosses the network interface it is monitoring and compares it against signatures or patterns of known malicious traffic to identify suspicious network traffic. If the packets match a signature, then the IDS takes the action defined in the rule such as logging the traffic in an alert file (Davidoff and Ham, 2012; Hagen, 2015a). The alerts produced by the IDS can be valuable to the analyst. They may provide a lead which will help the incident responder identify suspicious traffic and allow him to focus his investigation.

The presence of an alert does not necessarily mean that it that it is an incident requiring attention. A Microsoft IIS exploit targeting an Apache server may or might not be considered an event of interest. Nor does the absence of an alert necessarily mean that there is no malicious network traffic. It only means that no traffic that matches the signatures that were being checked. This would be termed a false negative.

2.7. Arpwatch

A computer needs to know the physical address (MAC address) of the network interface card of the destination system in order to communicate over the network. Mapping of the IP address to the MAC address is accomplished using the Address Resolution Protocol (ARP). When a computer does not know the MAC address, it broadcasts an ARP request asking who owns an IP address. The owner of the IP address responds with a unicast message indicating they own the IP address and provides the MAC address of the network interface to which they will receive traffic addressed to that IP address. Arpwatch monitors network traffic for ARP traffic and logs new IP/MAC address pairings and changes in IP/MAC address pairings to syslog. These log entries can be a good source of answering the questions such as when did a system first appear on a network? Were there unusual changes to IP/MAC address pairings that might

indicate an ARP spoofing attack? Was there a significant, unexpected increase in the number of new IP/MAC address pairings indicating a potential MAC flooding attempt?

The information logged by Arpwatch should corroborate the information captured by the DHCP logs. Both identify when a system appears on the network based on different transactions. Arpwatch looks for ARP requests and replies while DHCP looks for DHCP lease requests. Arpwatch also captures static IP/MAC address pairings while DHCP does not.

2.8. Netflow

Netflow collects a summary of the packets that are flowing across the network. Some of the key data captured includes the date and time, source and destination IP addresses, source and destination ports, protocol, the number of packets, and the amount of data transferred. From netflow data, the analyst can construct a detailed portrait of network activity (Hagen 2015b; Davidoff and Ham, 2012). By looking at who talked to who netflow data can be used to identify compromised hosts. By looking at data transfer volumes, it can help identify data leakage. Netflow data can help an analyst identify specific targets to help focus their investigation. It can be used as an index into full packet captures. Nfdump is an example of open-source netflow suite of tools that collect and processes netflow data (Hagen, 2015b; Haag, 2015.

2.9. Full Packet Capture

Full packet captures are valuable in that they provide a complete picture of what crossed the network. From packet captures, an analyst can reconstruct what happened on the network (Davidoff and Ham, 2012). Ideally, the analyst has access to the full packets for the period of time covering the event. This is not always the case.

There are challenges with full packet captures. The volume of data in packet captures can be considerable and as such, they will contain a lot of noise. A challenge is how to identify the traffic of importance and how to reduce the distraction of the noise. This is where other tools like netflow, DNS logs, DHCP logs, and IDS alerts can provide assistance. Each of these can provide leads on where to look. These leads may enable the analyst to focus on specific traffic, thus allowing for data reduction to reduce the

volume of data to examine and filtering out some of the noise. Another challenge is the storage of the large amounts of data collected. Full packet captures may only be retained for a short period of time. Tcpdump is a very popular packet capture tool included in Linux.

2.10. Syslog

Syslog is the standard logging mechanism for UNIX and Linux. It is the collection point for a number of log types like the DHCP logs. Based on the syslog configuration, all of the log data can be consolidated into a single log file or divided into multiple log files. Syslog also provides the option to send log entries to a central logging server.

An advantage of sending the syslog data to a central logging server is that it can be difficult and time-consuming to collect separate log files from many different servers. Centralizing the logging can simplify correlating log entries from multiple sources in a single location. Separating the logging from the server where the logging took place provides a layer of security by preventing an attacker from being able to edit the log files on the system that has been compromised to remove traces of their attack (Davidoff and Ham, 2012).

The incident responder needs to understand the logging architecture of the network in order to locate the log files. The configuration files can provide insight into this architecture. They can also point out deviations from the architecture.

3. Lab Setup

This section describes the basic architecture of the test environment. The Appendix contains detailed information about the installation and configuration of the various software components of the architecture.

3.1. The Architecture

The lab environment connects to the Internet through a router that acts as the Internet gateway with an IP address of 192.168.1.1 (see Figure 1). This router also

provides Internet DNS services. The Firewall/Router separates the internal network from the Internet Gateway/Router. All traffic between the internal network and the Internet must pass through this device. Traffic from the firewall/router accesses the internal network through a switch. This switch is a managed switch that permits port mirroring. A monitoring server is set up to collect network information. It captures network traffic from a span port on the switch and has visibility for all traffic going to and from the internal network and the Firewall/Router.

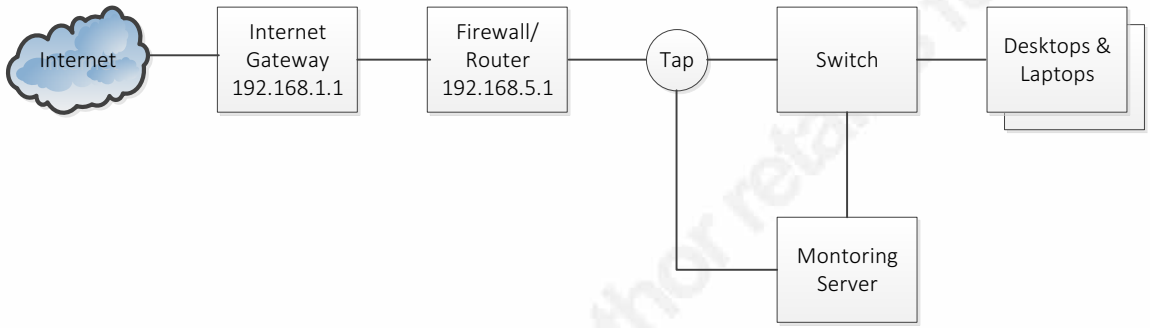

Figure 1: Physical Network Structure

The Internet Gateway is a wireless router provided by the ISP. The Firewall/Router System and the Monitoring System are physical Linux boxes running Centos 7.2 with a static IP address assigned to them. Since this is a lab/home network, the software is distributed between these two systems.

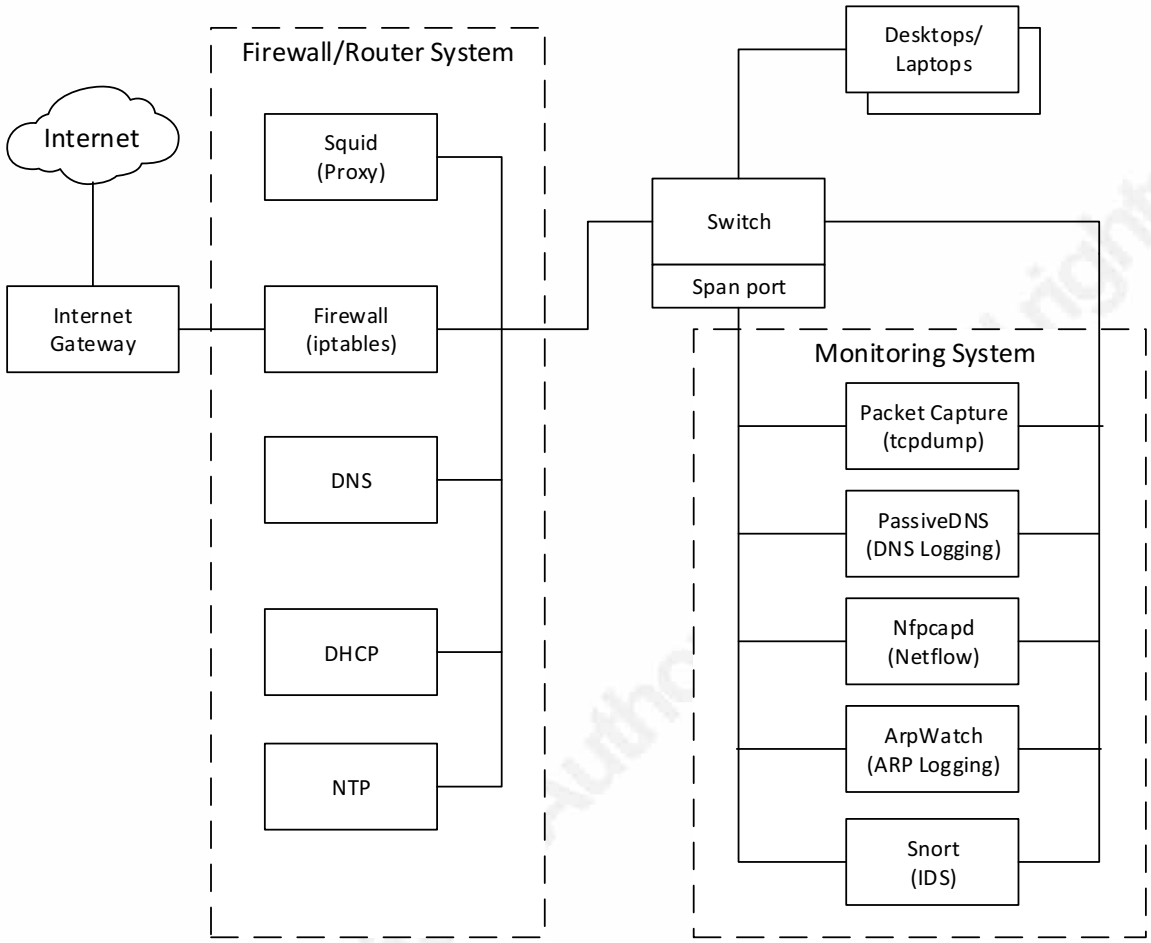

Figure 2: Logical Architecture

Figure 2 shows the logical architecture. The Firewall/Router System in addition to serving as a firewall/router, also hosts the internal network's NTP server, DHCP server, DNS server, and Squid proxy server. The Monitoring System hosts the full packet capture (tcpdump), the IDS server (Snort), the DNS logging server (PassiveDNS), ARPWatch, and Netflow server (nfpcapd).

4. Validation of the Network Artifact Architecture

A common initiator of incident response activities is an indicator that something unusual may be occurring. The initiator could be the notification that a large quantity of data was recently transferred to an external party. It could be an alert from the intrusion detection system. It could result from something usual seen in a log. To validate the

architecture supports network forensics during incident response, an incident was simulated, and then the data analyzed to determine that there was sufficient information to reconstruct the event forensically.

4.1. The Scenario

Given the importance of DNS, it is a best practice not to outsource DNS to a server on the Internet. Instead, a small number of internal DNS servers should reside on the internal network. DNS resolution would be directed to internal DNS servers and clients blocked from accessing external DNS servers (Hagen, 2015c). Internal DNS servers forward DNS queries to external DNS servers, when necessary, on behalf of the clients. In order to initiate our scenario two custom Snort rules were written to detect when someone tries to do a DNS query directly against an Internet-based DNS system rather than an internal one. The Snort rules are.

```
alert udp 192.168.5.0/24 any -> !192.168.5.1 53 (msg:"DNS query to external
DNS";sid:1000001;rev:0;)

alert tcp 192.168.5.0/24 any -> !192.168.5.1 53 (msg:"DNS query to external
DNS";sid:1000002;rev:0;)
```

The following activities were conducted to generate the event. The task is to analyze the forensic data and be able to reconstruct this activity. This was done by:

- Attaching a computer to the internal network, which has never been on the network before, on August 25, 2016, at 19:40 EST;
- Executing a DNS query directed to the external DNS server located at 8.8.8.8 by entering: "nslookup sra.com 8.8.8.8" in a cmd window at 19:52 to trigger our custom Snort rules and generate a Snort alert;
- Connecting to the euspba.org (Eastern US Pipe Band Association) website at 19:53 using a Firefox browser, version 47;
- Displaying the Voice page, located at http://euspba.org/thevoice.aspx, by selecting the Voice icon;

- Downloading the latest copy of the EUSPBA quarterly magazine, The Voice, in PDF format by clicking on the download option displayed a PDF file, http://euspba.org/voice/voice2016q2.pdf in the browser;
- Disconnecting the laptop from the network.

4.2. The Analysis

Two events occurred that triggered alerts with applications monitoring the network. Arpwatch sent two emails indicating a new system appeared on the network. The emails are:

```
From: root@serpent.fraser.local (Arpwatch)
To: root@serpent.fraser.local
Subject: new station
Message-Id: <20160825234023.C21D42000447@serpent.fraser.local>
Date: Thu, 25 Aug 2016 19:40:23 -0400 (EDT)

            hostname: <unknown>
          ip address: 169.254.111.154
    ethernet address: 8:d4:c:45:58:f3
     ethernet vendor: <unknown>
           timestamp: Thursday, August 25, 2016 19:40:23 -0400

From: root@serpent.fraser.local (Arpwatch)
To: root@serpent.fraser.local
Subject: new station
Message-Id: <20160825234026.E7DD92000447@serpent.fraser.local>
Date: Thu, 25 Aug 2016 19:40:26 -0400 (EDT)

            hostname: <unknown>
          ip address: 192.168.5.28
    ethernet address: 8:d4:c:45:58:f3
     ethernet vendor: <unknown>
           timestamp: Thursday, August 25, 2016 19:40:26 -0400
```

In addition to the Arpwatch alerts, five alerts appeared in the Snort alert log, /var/log/snort/alert, triggered by one of the custom Snort rules that looked for DNS queries bypassing the local DNS server and going directly to an external DNS server. The output of the alert log for this event is listed below. Of particular interest is that both sets of alerts are associated with the same IP address, 192.168.5.28.

```
[**] [1:1000001:0] DNS query to external DNS [**]
```

```
[Priority: 0]
08/25-19:52:15.308135 192.168.5.28:55500 -> 8.8.8.8:53
UDP TTL:128 TOS:0x0 ID:19969 IpLen:20 DgmLen:66
Len: 38

[**] [1:1000001:0] DNS query to external DNS [**]
[Priority: 0]
08/25-19:52:15.370231 192.168.5.28:55501 -> 8.8.8.8:53
UDP TTL:128 TOS:0x0 ID:19970 IpLen:20 DgmLen:66
Len: 38

[**] [1:1000001:0] DNS query to external DNS [**]
[Priority: 0]
08/25-19:52:15.393163 192.168.5.28:55502 -> 8.8.8.8:53
UDP TTL:128 TOS:0x0 ID:19971 IpLen:20 DgmLen:66
Len: 38

[**] [1:1000001:0] DNS query to external DNS [**]
[Priority: 0]
08/25-19:52:15.415642 192.168.5.28:55503 -> 8.8.8.8:53
UDP TTL:128 TOS:0x0 ID:19972 IpLen:20 DgmLen:53
Len: 25

[**] [1:1000001:0] DNS query to external DNS [**]
[Priority: 0]
08/25-19:52:15.441037 192.168.5.28:55504 -> 8.8.8.8:53
UDP TTL:128 TOS:0x0 ID:19973 IpLen:20 DgmLen:53
Len: 25
```

The first step in the analysis is to verify that NTP time services are running and that the time is synchronized between servers. As previously mentioned, it is important to verify the synchronization of timestamps in various log files and packet captures. The output below is from the internal NTP server, firefly.fraser.local, indicating that it is synchronizing with a stratus 2 time server, at 199.188.48.60. The time is synchronized to within 77 ms.

```
# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*tick.mdacore.ne 130.207.244.240  2 u  707 1024  377   20.830   -2.332   2.959
+fairy.mattnordh 200.98.196.212   2 u  706 1024  377   19.826    0.285   0.852
+a1.pcloud.com   200.98.196.212   2 u  686 1024  377   43.370   -1.566   0.617
-1.time.dbsinet. 64.113.32.5      2 u  398 1024  377   40.542    4.117   2.247

# ntpstat
synchronised to NTP server (199.188.48.60) at stratum 3
```

```
   time correct to within 77 ms
   polling server every 1024 s
```

The second output is from the monitoring system. It indicates that the server is synchronizing with the server NTP server, firefly.fraser.local, is a stratum 3 time server. Time is synchronized to within 107 ms.

```
# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*firefly.fraser. 66.228.59.187    3 u 1021 1024  377    0.170   -1.003   0.355

# ntpstat
synchronised to NTP server (192.168.5.1) at stratum 4
   time correct to within 107 ms
   polling server every 1024 s
```

By examining the Arpwatch log, /var/log/messages, we learn that this is the first time that Arpwatch has seen the IP Address 192.168.5.28. This corresponds with what was reported in the emails. The new computer appeared on the network August 25, 2016 at 19:40.

```
# grep arpwatch /var/log/messages | grep 8:d4:c:45:58:f3
Aug 25 19:40:21 serpent arpwatch: changed ethernet address 0.0.0.0
8:d4:c:45:58:f3 (30:5a:3a:d:99:6a)
Aug 25 19:40:23 serpent arpwatch: new station 169.254.111.154 8:d4:c:45:58:f3
Aug 25 19:40:26 serpent arpwatch: new station 192.168.5.28 8:d4:c:45:58:f3
```

Because of our knowledge of our environment, we know that the IP address of the suspect computer falls within the range assigned by the DHCP server. Relevant information from the DHCP configuration file, /etc/dhcp/dhcpd.conf, is:

```
default-lease-time 86400;    # time in seconds - 1 days
max-lease-time 259200;       # time in seconds - 3 days
authoritative;
log-facility local7;

subnet 192.168.5.0 netmask 255.255.255.0 {
  range 192.168.5.26 192.168.5.99;
  option routers 192.168.5.1;
  option broadcast-address 192.168.5.255;
}
```

Knowing that DHCP is involved leads us to check out DHCP logging, which is found in /var/log/messages.

```
# grep 192.168.5.28 /var/log/messages | grep dhcpd
Aug 25 19:40:25 firefly dhcpd: DHCPOFFER on 192.168.5.28 to 08:d4:0c:45:58:f3
(octopus) via enp5s2
Aug 25 19:40:25 firefly dhcpd: DHCPREQUEST for 192.168.5.28 (192.168.5.1) from
08:d4:0c:45:58:f3 (octopus) via enp5s2
Aug 25 19:40:25 firefly dhcpd: DHCPACK on 192.168.5.28 to 08:d4:0c:45:58:f3
(octopus) via enp5s2
```

The log output from the presence of the DHCPOFFER directive indicates that the computer requested a DHCP lease. This is confirmed by examining the log file using the computer's MAC address. All four steps in the DHCP handshake requesting an IP address – DHCPDISCOVER, DHCOFFER, DHCPRQUEST, and DHCPACK – are present. This is further evidence indicating that the computer joined the network August 25, 2016 at 19:40.

```
# grep 08:d4:0c:45:58:f3 /var/log/messages | grep dhcpd
Aug 25 19:40:24 firefly dhcpd: DHCPDISCOVER from 08:d4:0c:45:58:f3 via enp5s2
Aug 25 19:40:25 firefly dhcpd: DHCPOFFER on 192.168.5.28 to 08:d4:0c:45:58:f3
(octopus) via enp5s2
Aug 25 19:40:25 firefly dhcpd: DHCPREQUEST for 192.168.5.28 (192.168.5.1) from
08:d4:0c:45:58:f3 (octopus) via enp5s2
Aug 25 19:40:25 firefly dhcpd: DHCPACK on 192.168.5.28 to 08:d4:0c:45:58:f3
(octopus) via enp5s2
```

We also know from the log files that the computer did not remain on the network for more than 12 hours. We know this because a computer will initiate the renewal of its lease halfway through the lease period, which in this case is 12 hours. Since there are no lease renewal transactions – DHCPREQUEST, DHCPACK – we can conclude the computer is no longer on the network.

So far we know from the log files that a computer appeared on the network August 25 at 19:40. It requested an IP address from the DHCP server and was assigned 192.168.5.28. We also know, from the Snort alert, that a computer at that IP address contacted an external DNS server located at IP address 8.8.8.8 at 19:52.

To get a sense of what kind of traffic is flowing on the network from the suspect computer (192.168.5.28) three netflow queries were run. One summarized all traffic. One summarized UDP traffic. The other summarized TCP traffic. This can help to identify what traffic to examine.

```
# nfdump -R /var/log/netflow -O bytes -t '2016/08/25.14:00:00-2016/08/25.23:00'
-A proto -o 'fmt:%ts %te %pr %byt %fl' 'ip 192.168.5.28'
Date first seen          Date last seen          Proto    Bytes Flows
2016-08-25 19:40:28.533 2016-08-25 19:58:07.574 TCP     25.0 M   215
2016-08-25 19:40:25.266 2016-08-25 19:56:32.253 UDP      74230   266
2016-08-25 19:40:26.892 2016-08-25 19:40:26.892 ICMP        28     1
Summary: total flows: 482, total bytes: 25076401, total packets: 11565, avg
bps: 188844, avg pps: 10, avg bpp: 2168
Time window: 2016-08-25 14:00:00 - 2016-08-25 21:24:58
Total flows processed: 2357, Blocks skipped: 0, Bytes read: 119736
Sys: 0.007s flows/second: 299035.8   Wall: 0.004s flows/second: 518135.9

# nfdump -R /var/log/netflow -O flows -t '2016/08/25.14:00-2016/08/25.23:59' -A
srcip,dstport -o 'fmt:%sa -> %dp %byt %fl' 'proto udp and src ip 192.168.5.28'
     Src IP Addr    Dst Pt    Bytes Flows
    192.168.5.28 ->     53     6551   129
    192.168.5.28 ->   5355     1059    18
    192.168.5.28 ->   3289      132     6
    192.168.5.28 ->   3702    17392     5
    192.168.5.28 ->   1900     2298     1
    192.168.5.28 ->    137     2446     1
Summary: total flows: 160, total bytes: 29878, total packets: 246, avg bps:
247, avg pps: 0, avg bpp: 121
Time window: 2016-08-25 14:00:00 - 2016-08-25 21:34:58
Total flows processed: 2386, Blocks skipped: 0, Bytes read: 121176
Sys: 0.009s flows/second: 265022.8   Wall: 0.005s flows/second: 442589.5

# nfdump -R /var/log/netflow -O flows -t '2016/08/25.14:00-2016/08/25.23:59' -A
srcip,dstport -o 'fmt:%sa -> %dp %byt %fl' 'proto tcp and src ip 192.168.5.28'
     Src IP Addr    Dst Pt    Bytes Flows
    192.168.5.28 ->    443   406528    61
    192.168.5.28 ->     80    84894    45
    192.168.5.28 ->     53      162     1
Summary: total flows: 107, total bytes: 491584, total packets: 3711, avg bps:
3976, avg pps: 3, avg bpp: 132
Time window: 2016-08-25 14:00:00 - 2016-08-25 21:34:58
Total flows processed: 2386, Blocks skipped: 0, Bytes read: 121176
Sys: 0.008s flows/second: 268240.6   Wall: 0.005s flows/second: 469500.2
```

The netflow analysis identifies three protocols associated with the suspect computer. UDP and TCP constituted the majority of the traffic. There was only one

ICMP packet. Of the UDP traffic, the bulk of the traffic is on port 53. This is generally DNS traffic. The TCP traffic is primarily on port 443, which is generally HTTPS traffic and on port 80, which is generally HTTP traffic. There is one instance of TCP traffic on port 53. Based on this information we can focus our investigation on DNS, HTTP, and HTTPS traffic.

In order to reduce the flow data to only the traffic of interest, the following extracts were made. One for all traffic associated with 192.1685.28 – nfcapd.20160825-28. One with only HTTP traffic – nfcapd.20160825-p80. And one with only HTTPS traffic – nfcapd.201600825-p443.

```
# nfdump -R /var/log/netflow -t '2016/08/25.14:00-2016/08/25.23:59' -w
/tmp/nfcapd.20160825 'proto tcp and ip 192.168.5.28'

# nfdump -r nfcapd.20160825-28 -w nfcapd.20160825-p80 'port 80'

# nfdump -r nfcapd.20160825-28 -w nfcapd.20160825-p443 'port 443'

# nfdump -r nfcapd.20160825-p80 -O bytes -A srcip,dstip -o 'fmt:%sa %da %byt
%pkt %fl'
     Src IP Addr      Dst IP Addr    Bytes  Packets Flows
  104.96.220.171     192.168.5.28   11.8 M     2705     3
    50.21.177.33     192.168.5.28    6.3 M     1993     7
  65.216.231.144     192.168.5.28    2.5 M      815    16
    192.168.5.28     50.21.177.33    24664      583     7
    192.168.5.28   104.96.220.171    21039     1015     3
    192.168.5.28   65.216.231.144    20609      345    16
     72.21.91.29     192.168.5.28     8668       41     2
    192.168.5.28    65.52.108.153     6333        9     1
    192.168.5.28      72.21.91.29     4291       39     2
    65.52.108.27     192.168.5.28     2674       11     2
  216.58.217.142     192.168.5.28     2458       23     3
    192.168.5.28     65.52.108.27     2238       15     2
    192.168.5.28   143.127.93.106     1501       11     4
    192.168.5.28   216.58.217.142     1403       25     3
   65.52.108.153     192.168.5.28      991        9     1
  143.127.93.106     192.168.5.28      956       10     4
    192.168.5.28   63.245.201.111      944        8     2
  63.245.201.111     192.168.5.28      926        9     2
   65.222.200.80     192.168.5.28      908       14     1
   131.253.61.68     192.168.5.28      767        5     1
  104.90.101.179     192.168.5.28      644        3     1
    192.168.5.28   63.245.197.112      518        5     1
  63.245.197.112     192.168.5.28      494        6     1
  104.96.221.145     192.168.5.28      463        4     1
```

```
    192.168.5.28    131.253.61.68      463        6     1
    192.168.5.28   104.96.221.145      418        6     1
    192.168.5.28    65.222.200.80      239        4     1
    192.168.5.28   104.90.101.179      234        5     1
Summary: total flows: 90, total bytes: 20671464, total packets: 7724, avg bps:
156152, avg pps: 7, avg bpp: 2676
Time window: 2016-08-25 19:40:28 - 2016-08-25 19:58:07
Total flows processed: 90, Blocks skipped: 0, Bytes read: 4344
Sys: 0.005s flows/second: 17314.4    Wall: 0.002s flows/second: 30395.1


# nfdump -r nfcapd.20160825-p443 -O bytes -A srcip,dstip -o 'fmt:%sa %da %byt
%pkt %fl'
     Src IP Addr      Dst IP Addr    Bytes  Packets Flows
  216.58.195.142     192.168.5.28    3.1 M     1025     2
    54.192.55.95     192.168.5.28   319009      116     2
    192.168.5.28   134.170.58.189   259192      106     2
    54.192.55.77     192.168.5.28   124020       56     2
    31.13.69.228     192.168.5.28   123916       60     1
    31.13.69.203     192.168.5.28    35530       25     1
   54.191.11.118     192.168.5.28    30431       22     1
    192.168.5.28   216.58.195.142    29335      844     2
    172.217.1.14     192.168.5.28    22610       44     1
    192.168.5.28  173.194.207.136    22560       30     2
      64.4.27.50     192.168.5.28    21542       23     2
  63.245.192.201     192.168.5.28    21175       47     6
     166.98.6.85     192.168.5.28    19836       40     4
    192.168.5.28       64.4.27.50    16429       23     2
  216.10.195.252     192.168.5.28    13022       22     3
   131.253.61.68     192.168.5.28    12988       10     1
   54.200.62.216     192.168.5.28    10765       28     2
 173.194.207.136     192.168.5.28    10757       28     2
  143.127.102.98     192.168.5.28    10598       25     3
  134.170.58.189     192.168.5.28    10212      101     2
    192.168.5.28      166.98.6.85    10164       40     4
    192.168.5.28   63.245.192.201    10146       40     6
    52.72.67.195     192.168.5.28     8960        9     1
    54.192.55.44     192.168.5.28     8387       21     2
    192.168.5.28   216.10.195.252     7116       23     3
   157.55.240.89     192.168.5.28     5988        9     2
  104.90.101.179     192.168.5.28     5479        6     1
    192.168.5.28     52.34.127.11     5275       17     2
    65.52.108.29     192.168.5.28     5240        8     1
     166.98.6.31     192.168.5.28     4903       11     1
  143.127.102.34     192.168.5.28     4823        9     1
    23.79.197.69     192.168.5.28     4583        6     1
    65.55.44.109     192.168.5.28     4431        7     1
    65.55.252.93     192.168.5.28     4400        6     1
    192.168.5.28     65.52.108.29     4095       10     1
```

```
   54.187.236.54     192.168.5.28     3978       13     1
   50.112.202.19     192.168.5.28     3857       13     1
    52.34.127.11     192.168.5.28     3836       15     2
  54.201.247.238     192.168.5.28     3792       13     1
    192.168.5.28    131.253.61.68     3744       13     1
    192.168.5.28     65.55.252.93     3404       10     1
    192.168.5.28     172.217.1.14     3326       31     1
    192.168.5.28     31.13.69.228     2789       50     1
    192.168.5.28     31.13.69.203     2662       30     1
    192.168.5.28    54.200.62.216     2549       33     2
    192.168.5.28   143.127.102.98     2440       25     3
    192.168.5.28    157.55.240.89     2307       14     1
    192.168.5.28     54.192.55.95     2220       70     2
    192.168.5.28     65.55.44.109     2158       10     1
    192.168.5.28   143.127.102.34     2010        8     1
    192.168.5.28     54.192.55.77     1754       49     2
    192.168.5.28      166.98.6.31     1581       10     1
    192.168.5.28     54.192.55.44     1292       23     2
    192.168.5.28    54.191.11.118     1290       19     1
    192.168.5.28    54.187.236.54     1213       15     1
    192.168.5.28   54.201.247.238     1155       14     1
    192.168.5.28    50.112.202.19     1044       15     1
    192.168.5.28   157.56.141.114      920       30    10
    192.168.5.28     23.79.197.69      813        8     1
    192.168.5.28     52.72.67.195      785       11     1
    192.168.5.28   104.90.101.179      760        9     1
  157.56.141.114     192.168.5.28      600       30    10
Summary: total flows: 123, total bytes: 4329353, total packets: 3478, avg bps:
35040, avg pps: 3, avg bpp: 1244
Time window: 2016-08-25 19:40:28 - 2016-08-25 19:56:57
Total flows processed: 123, Blocks skipped: 0, Bytes read: 5928
Sys: 0.005s flows/second: 22875.2    Wall: 0.003s flows/second: 39626.3
```

Digging deeper into the suspected DNS traffic using a netflow query shows that most of the traffic, 225 flows, were between the computer of interest and the internal DNS server (192.168.5.1). Five flows, which correspond with the five Snort alerts were with the external DNS server (8.8.8.8). The time of these queries matches the timestamp of the Snort alert. Furthermore, we also know that the first transaction to the internal DNS server occurred at 19:40:26 and the last transaction occurred at 19:54:09.

```
# nfdump -R /var/log/netflow -O tstart -t '2016/08/25.19:00-2016/08/25.20:00' -
A srcip,dstip,proto -o 'fmt:%ts %sa %da %pr %byt %fl' 'ip 192.168.5.28 and port
53'
Date first seen             Src IP Addr      Dst IP Addr Proto    Bytes Flows
```

```
2016-08-25 19:40:26.926     192.168.5.28      192.168.5.1 UDP       6347   124
2016-08-25 19:40:26.926      192.168.5.1     192.168.5.28 UDP      43266   100
2016-08-25 19:52:15.308     192.168.5.28          8.8.8.8 UDP        204     5
2016-08-25 19:52:15.338          8.8.8.8     192.168.5.28 UDP        470     5
2016-08-25 19:54:09.627      192.168.5.1     192.168.5.28 TCP       1164     1
2016-08-25 19:54:09.627     192.168.5.28      192.168.5.1 TCP        162     1
Summary: total flows: 236, total bytes: 51613, total packets: 243, avg bps:
427, avg pps: 0, avg bpp: 212
Time window: 2016-08-25 19:00:02 - 2016-08-25 19:59:59
Total flows processed: 1196, Blocks skipped: 0, Bytes read: 59856
Sys: 0.011s flows/second: 107139.7   Wall: 0.006s flows/second: 187784.6
```

Since the passiveDNS log file stores the date in UNIX epoch time format, it is advisable to preprocess the log file to convert the dates to a human readable format. The following command does this while also filtering out records not associated with the computer of interest (Hagen, 2015d). An MD5 hash was taken of the converted file.

```
# grep 192.168.5.28 20160825_0001.log | sed 's/||/|/g' | awk -F'|' '{OFS="|";
printf("%s",strftime("%Y-%m-%d_%H:%M:%S",$1));$1=""; print $0}' >
/tmp/passivedns-20160825-humanreadable.log

# md5sum passivedns-20160825-humanreadable.log
15db53575771968062662cc52a972377  passivedns-20160825-humanreadable.log
```

From the DNS log, we confirm that the traffic to 8.8.8.8 is DNS traffic. The timestamp in the log file matches the timestamp in the Snort alert. We also identify the request is for the IP address of sra.com. The response returned by DNS was 163.252.95.35.

```
# grep 8.8.8.8 passivedns-20160825-humanreadable.log
2016-08-25_19:52:15|192.168.5.28|8.8.8.8|IN|8.8.8.8.in-addr.arpa.|PTR|google-
public-dns-a.google.com.|21599|1
2016-08-25_19:52:15|192.168.5.28|8.8.8.8|IN|sra.com.|A|163.252.95.35|2930|1
```

Next we query the netflow data to determine what traffic was exchanged with the IP address 163.252.95.35. Our query does not turn up any traffic.

```
# nfdump -R /var/log/netflow -O tstart -t '2016/08/25.19:00-2016/08/25.20:00' -
A srcip,dstip,proto -o 'fmt:%ts %sa %dap %byt %fl' 'ip 163.252.95.35'
Date first seen             Src IP Addr      Dst IP Addr:Port     Bytes Flows
Summary: total flows: 0, total bytes: 0, total packets: 0, avg bps: 0, avg pps:
0, avg bpp: 0
Time window: 2016-08-25 19:00:02 - 2016-08-25 19:59:59
Total flows processed: 1196, Blocks skipped: 0, Bytes read: 59856
Sys: 0.009s flows/second: 121249.0   Wall: 0.005s flows/second: 229426.4
```

The investigation might end here labeling the Snort alert as uninteresting. However, because this is an unknown computer, we will continue on a little further and examine the HTTP traffic.

A netflow query for port 80 traffic to and from our system of interest generates the following output. The two top talkers are 32 flows with 65.216.231.144 and 14 flows with 50.21.177.33. Also of note is that there are a large number of bytes transferred with 65.216.231.144 (2.5 M), 50.21.177.33 (6.3 M), 104.96.220.171 (11.8 M). This is a good place to start.

```
# nfdump -r nfcapd.20160825-p80 -O flows -A srcip,dstip -o 'fmt:%sa %da %byt
%pkt %fl'
     Src IP Addr      Dst IP Addr    Bytes  Packets Flows
  65.216.231.144     192.168.5.28    2.5 M      815    16
    192.168.5.28   65.216.231.144    20609      345    16
    192.168.5.28     50.21.177.33    24664      583     7
    50.21.177.33     192.168.5.28    6.3 M     1993     7
    192.168.5.28   143.127.93.106     1501       11     4
  143.127.93.106     192.168.5.28      956       10     4
  216.58.217.142     192.168.5.28     2458       23     3
  104.96.220.171     192.168.5.28   11.8 M     2705     3
    192.168.5.28   104.96.220.171    21039     1015     3
    192.168.5.28   216.58.217.142     1403       25     3
    192.168.5.28      72.21.91.29     4291       39     2
  63.245.201.111     192.168.5.28      926        9     2
    65.52.108.27     192.168.5.28     2674       11     2
     72.21.91.29     192.168.5.28     8668       41     2
    192.168.5.28     65.52.108.27     2238       15     2
    192.168.5.28   63.245.201.111      944        8     2
    192.168.5.28   63.245.197.112      518        5     1
    192.168.5.28    65.222.200.80      239        4     1
    192.168.5.28   104.96.221.145      418        6     1
  63.245.197.112     192.168.5.28      494        6     1
   131.253.61.68     192.168.5.28      767        5     1
   65.52.108.153     192.168.5.28      991        9     1
    192.168.5.28    131.253.61.68      463        6     1
  104.96.221.145     192.168.5.28      463        4     1
    192.168.5.28    65.52.108.153     6333        9     1
  104.90.101.179     192.168.5.28      644        3     1
   65.222.200.80     192.168.5.28      908       14     1
    192.168.5.28   104.90.101.179      234        5     1
Summary: total flows: 90, total bytes: 20671464, total packets: 7724, avg bps:
156152, avg pps: 7, avg bpp: 2676
Time window: 2016-08-25 19:40:28 - 2016-08-25 19:58:07
Total flows processed: 90, Blocks skipped: 0, Bytes read: 4344
```

```
Sys: 0.005s flows/second: 17367.8    Wall: 0.002s flows/second: 30110.4
```

The DNS logs provided the identity of the hosts at 65.216.231.144, 50.21.177.33, and 104.96.220.171.

```
# grep 50.21.177.33 passivedns-20160825-humanreadable.log
2016-08-
25_19:56:27|192.168.5.28|192.168.5.1|IN|euspba.org.|A|50.21.177.33|3600|1
2016-08-
25_19:56:28|192.168.5.28|192.168.5.1|IN|www.euspba.org.|A|50.21.177.33|3600|1
2016-08-
25_19:56:28|192.168.5.28|192.168.5.1|IN|www.euspba.org.|A|50.21.177.33|3600|1
2016-08-
25_19:56:27|192.168.5.28|192.168.5.1|IN|euspba.org.|A|50.21.177.33|3600|1

# grep 65.216.231.144 passivedns-20160825-humanreadable.log
2016-08-
25_19:51:01|192.168.5.28|192.168.5.1|IN|a568.d.akamai.net.|A|65.216.231.144|9|2

# grep 104.96.220.171 passivedns-20160825-humanreadable.log
2016-08-
25_19:43:03|192.168.5.28|192.168.5.1|IN|a1670.dspg2.akamai.net.|A|104.96.220.17
1|20|1
```

Additional insight can be obtained by looking at the HTTP host header in the full packet capture. The host header for 65.216.231.144 identifies this as Symantec liveupdate traffic.

```
# tshark -n -r 20160825_1901.pcap -T fields -e ip.src -e ip.dst -e http.host -Y
'http.request and ip.addr==192.168.5.28' | sort | uniq
192.168.5.28    104.90.101.179  go.microsoft.com
192.168.5.28    104.96.220.171  wscont.apps.microsoft.com.edgesuite.net
192.168.5.28    104.96.221.145  ctldl.windowsupdate.com
192.168.5.28    131.253.61.68   login.live.com
192.168.5.28    143.127.93.106  spoc-pool-gtm.norton.com
192.168.5.28    216.58.217.142  clients1.google.com
192.168.5.28    239.255.255.250 239.255.255.250:1900
192.168.5.28    50.21.177.33    euspba.org
192.168.5.28    63.245.197.112  stats.norton.com
192.168.5.28    63.245.201.111  stats.norton.com
192.168.5.28    65.216.231.144  liveupdate.symantecliveupdate.com
192.168.5.28    65.222.200.80   liveupdate.symantec.com
192.168.5.28    65.52.108.153   statsfe2.update.microsoft.com
192.168.5.28    65.52.108.27    g.bing.com
192.168.5.28    72.21.91.29     ocsp.digicert.com
```

Running an nslookup query on liveupdate.symantec.com confirms that this is an alias for a568.d.akamai.net. We will classify this traffic as normal and ignore it for this analysis.

```
>nslookup liveupdate.symantec.com
Server:  firefly.fraser.local
Address:  192.168.5.1

Non-authoritative answer:
Name:    a568.d.akamai.net
Addresses:  96.6.113.35
          96.6.113.8
Aliases:  liveupdate.symantec.com
          liveupdate.symantec.d4p.net
          symantec.georedirector.akadns.net
```

A Similar examination of the HTTP host header for 104.96.220.171 identifies the site as wscont.apps.microsoft.com.edgesuite.net. A nslookup confirms that this is an alias for a1670.dspg2.akamai.net. We will classify this traffic as normal and ignore it for our analysis.

```
C:\Users\fraserg>nslookup wscont.apps.microsoft.com.edgesuite.net
Server:  firefly.fraser.local
Address:  192.168.5.1

Non-authoritative answer:
Name:    a1670.dspg2.akamai.net
Addresses:  2600:1400:a::1743:faa1
          2600:1400:a::1743:faab
          104.96.221.90
          104.96.221.97
Aliases:  wscont.apps.microsoft.com.edgesuite.net
```

Next is the examination of the traffic with the IP address 50.21.177.33, euspba.org. The Squid access.log, shown below, can provide more information. The User Agent string identifies the browser as Firefox. The website was first accessed at 19:56:27 and last accessed at 19:56:39.

There were three pages displayed in this exchange – http://euspba.org/, http://euspba.org/thevoice.aspx, and http://euspba.org/voice/voice2016q2.pdf. The other HTTP request/responses were for objects on the pages. It would appear that

http://euspba.org was entered into the browser because there is no referer specified. The Squid result code of TCP_MISS indicates that the objects were not found in the Squid cache. The Hierarchy code of HIER_DIRECT indicates that the object was retrieved from the original server (Squid wiki, 2015).

Http://euspba.org/thevoice.aspx was accessed by clicking on a link of the http://euspba.org site. We know this because euspba.org is specified as the referrer. Likewise, http:// euspba.org/voice/voice2016q2.pdf was accessed by clicking on a link on the Http://euspba.org/thevoice.aspx since it is listed as the referer.

```
# grep euspba access.log
192.168.5.28 - - [25/Aug/2016:19:56:27 -0400] "GET http://euspba.org/ HTTP/1.1"
200 58570 "-" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0) Gecko/20100101
Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:27 -0400] "GET http://euspba.org/_style.css
HTTP/1.1" 200 6282 "http://euspba.org/" "Mozilla/5.0 (Windows NT 6.3; WOW64;
rv:47.0) Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:27 -0400] "GET
http://euspba.org/WebResource.axd? HTTP/1.1" 200 23533 "http://euspba.org/"
"Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0"
TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:27 -0400] "GET
http://euspba.org/WebResource.axd? HTTP/1.1" 200 27421 "http://euspba.org/"
"Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0"
TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET http://euspba.org/DXR.axd?
HTTP/1.1" 200 51895 "http://euspba.org/" "Mozilla/5.0 (Windows NT 6.3; WOW64;
rv:47.0) Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET
http://euspba.org/images/logo_euspba.gif HTTP/1.1" 200 2391
"http://euspba.org/" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0)
Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET http://euspba.org/DXR.axd?
HTTP/1.1" 200 484 "http://euspba.org/" "Mozilla/5.0 (Windows NT 6.3; WOW64;
rv:47.0) Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET
http://euspba.org/images/logo_euspba_text_bk.gif HTTP/1.1" 200 3212
"http://euspba.org/" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0)
Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET
http://euspba.org/images/thevoice.jpg HTTP/1.1" 200 24158 "http://euspba.org/"
"Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0"
TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET
http://euspba.org/images/bkgd_body.jpg HTTP/1.1" 200 1416
"http://euspba.org/_style.css" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0)
Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET
http://euspba.org/App_Themes/BlackGlass/GridView/Loading.gif HTTP/1.1" 200
10962 "http://euspba.org/" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0)
Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
```

```
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET
http://euspba.org/images/bkgd_site.jpg HTTP/1.1" 200 2541
"http://euspba.org/_style.css" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0)
Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET http://euspba.org/DXR.axd?
HTTP/1.1" 200 213048 "http://euspba.org/" "Mozilla/5.0 (Windows NT 6.3; WOW64;
rv:47.0) Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET http://euspba.org/DXR.axd?
HTTP/1.1" 200 585
"http://euspba.org/DXR.axd?r=1_10,1_12,1_1,0_413,0_570,0_415,0_574,0_406,1_5,0_
408-ukyRc" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0) Gecko/20100101
Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET http://euspba.org/DXR.axd?
HTTP/1.1" 200 8338
"http://euspba.org/DXR.axd?r=1_10,1_12,1_1,0_413,0_570,0_415,0_574,0_406,1_5,0_
408-ukyRc" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0) Gecko/20100101
Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET http://euspba.org/DXR.axd?
HTTP/1.1" 200 16539
"http://euspba.org/DXR.axd?r=1_10,1_12,1_1,0_413,0_570,0_415,0_574,0_406,1_5,0_
408-ukyRc" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0) Gecko/20100101
Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:28 -0400] "GET
http://euspba.org/favicon.ico HTTP/1.1" 404 5703 "-" "Mozilla/5.0 (Windows NT
6.3; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:29 -0400] "GET
http://euspba.org/favicon.ico HTTP/1.1" 404 5703 "-" "Mozilla/5.0 (Windows NT
6.3; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT

192.168.5.28 - - [25/Aug/2016:19:56:31 -0400] "GET
http://euspba.org/thevoice.aspx HTTP/1.1" 200 21552 "http://euspba.org/"
"Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0"
TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:31 -0400] "GET http://euspba.org/DXR.axd?
HTTP/1.1" 200 21823 "http://euspba.org/thevoice.aspx" "Mozilla/5.0 (Windows NT
6.3; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:31 -0400] "GET http://euspba.org/DXR.axd?
HTTP/1.1" 200 84106 "http://euspba.org/thevoice.aspx" "Mozilla/5.0 (Windows NT
6.3; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
192.168.5.28 - - [25/Aug/2016:19:56:31 -0400] "GET
http://euspba.org/images/logo_voice.jpg HTTP/1.1" 200 18580
"http://euspba.org/thevoice.aspx" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0)
Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT

192.168.5.28 - - [25/Aug/2016:19:56:38 -0400] "GET
http://euspba.org/voice/voice2016q2.pdf HTTP/1.1" 206 33774
"http://euspba.org/thevoice.aspx" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0)
Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT

192.168.5.28 - - [25/Aug/2016:19:56:39 -0400] "GET
http://euspba.org/voice/voice2016q2.pdf HTTP/1.1" 200 5604277
"http://euspba.org/thevoice.aspx" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0)
Gecko/20100101 Firefox/47.0" TCP_MISS:HIER_DIRECT
```

The Squid access log mirrors the information captured in the full packet capture as reported by tshark.

Gordon Fraser, Gordon.fraser@ctipc.com

```
# tshark -n -r 20160825_1901.pcap -T fields -e frame.time -e
http.request.method -e http.host -e http.request.uri -Y 'http.request and
ip.addr==192.168.5.28' | grep euspba
Running as user "root" and group "root". This could be dangerous.
"Aug 25, 2016 19:56:27.433348000 EDT"   GET     euspba.org      /
"Aug 25, 2016 19:56:27.708645000 EDT"   GET     euspba.org
/DXR.axd?r=1_10,1_12,1_1,0_413,0_570,0_415,0_574,0_406,1_5,0_408-ukyRc
"Aug 25, 2016 19:56:27.709414000 EDT"   GET     euspba.org      /_style.css
"Aug 25, 2016 19:56:27.709615000 EDT"   GET     euspba.org
/WebResource.axd?d=sK_ws0WXRC8s0oBVsdV_B9Cyb_sn0TckvwP-
lHaCpd2tTzMyWEFofPoSgV5KBr6jz9-qf4Lts4t_pr3PE_fx-
TUudAbAd3vw0E0z8tDcg4M1&t=635823490080000000
"Aug 25, 2016 19:56:27.711276000 EDT"   GET     euspba.org
/WebResource.axd?d=PI_DeH2Sjjs-
iUkarBhiXDo9uHb4GwVWue0iZIFIqDvw5KV8TsPMRnBmU0S1YicA2PlDPLoYdAgJYqSnpfsFIRKZQyM
9GuuiI8B-zgf2Eqs1&t=635823490080000000
"Aug 25, 2016 19:56:27.712465000 EDT"   GET     euspba.org
/DXR.axd?r=1_171,1_94,1_164,1_91,1_156,1_162,1_147,1_104,1_138,1_114,1_121,1_11
3,1_98,1_154,1_106,1_152,1_97,1_150-ukyRc
"Aug 25, 2016 19:56:28.023660000 EDT"   GET     euspba.org
/images/logo_euspba.gif
"Aug 25, 2016 19:56:28.023969000 EDT"   GET     euspba.org
/images/logo_euspba_text_bk.gif
"Aug 25, 2016 19:56:28.024481000 EDT"   GET     euspba.org
/DXR.axd?r=1_15-ukyRc
"Aug 25, 2016 19:56:28.024498000 EDT"   GET     euspba.org
/App_Themes/BlackGlass/GridView/Loading.gif
"Aug 25, 2016 19:56:28.026324000 EDT"   GET     euspba.org
/images/thevoice.jpg
"Aug 25, 2016 19:56:28.077885000 EDT"   GET     euspba.org
/images/bkgd_body.jpg
"Aug 25, 2016 19:56:28.087142000 EDT"   GET     euspba.org
/images/bkgd_site.jpg
"Aug 25, 2016 19:56:28.363643000 EDT"   GET     euspba.org
/DXR.axd?r=0_411-MjyRc
"Aug 25, 2016 19:56:28.365135000 EDT"   GET     euspba.org
/DXR.axd?r=0_414-MjyRc
"Aug 25, 2016 19:56:28.365152000 EDT"   GET     euspba.org
/DXR.axd?r=0_571-MjyRc
"Aug 25, 2016 19:56:28.903780000 EDT"   GET     euspba.org      /favicon.ico
"Aug 25, 2016 19:56:28.958335000 EDT"   GET     euspba.org      /favicon.ico
"Aug 25, 2016 19:56:31.357109000 EDT"   GET     euspba.org      /thevoice.aspx
"Aug 25, 2016 19:56:31.530683000 EDT"   GET     euspba.org
/DXR.axd?r=1_10,1_12,1_1-ukyRc
"Aug 25, 2016 19:56:31.532771000 EDT"   GET     euspba.org
/DXR.axd?r=1_171,1_94,1_164,1_91,1_156,1_162,1_147-ukyRc
"Aug 25, 2016 19:56:31.602186000 EDT"   GET     euspba.org
/images/logo_voice.jpg
"Aug 25, 2016 19:56:37.992256000 EDT"   GET     euspba.org
/voice/voice2016q2.pdf
"Aug 25, 2016 19:56:38.828108000 EDT"   GET     euspba.org
/voice/voice2016q2.pdf
```

The PDF file was extracted from the packet capture using Wireshark. This was done by locating the packet containing the PDF file using the filter: http.request.uri contains voice2016q2.pdf. On expanding the Hypertext Transfer Protocol section, a link was displayed labeled Request in frame: xxx. The display filter had to be cleared in order to make the response packet available. Then clicking on the link for response in frame xxx brings up the response record. The PDF file was exported by right-clicking on the Media type field and selecting the "Export Packet bytes" menu option. The file was saved as a raw file.

In order to verify that the file exported from the packet capture was indeed the file that was downloaded, an MD5 hash was done of the original file and compared to the MD5 hash of the file that was downloaded. As shown below, the MD5 hashes match. Sometimes when a file is exported from packet captures, the analyst does not know what type of file it is. The Linux command "file" can be used to identify the type of file.

```
[root@serpent fraserg]# file export.raw
export.raw: PDF document, version 1.7
[root@serpent fraserg]# md5sum export.raw
fc3ad7cb0825a65ee3321d629cd3d399  export.raw

[root@serpent fraserg]# file voice2016q2.pdf
voice2016q2.pdf: PDF document, version 1.7
[root@serpent fraserg]# md5sum voice2016q2.pdf
fc3ad7cb0825a65ee3321d629cd3d399  voice2016q2.pdf
```

4.3. The Results

Table 1 presents a timeline of the events that were discovered using the forensic information captured by our network architecture. Using the network artifacts captured by our network architecture we were able to reconstruct all of the network events accurately. These events matched the actions that were used to generate the test scenario. Traffic, we classified as normal, is not included in the table.

Table 1: Event Timeline (all events occurred on August 24, 2016)

Time (EDT)	Event	Source
2016-08-25 19:40:26	Received an email indicating a new system appeared on the network	Arpwatch
2016-08-25 19:40	Computer requested IP address from the DHCP server; obtained a lease on 192.18.5.28	DHCP server log
2016-08-25 19:40	Traffic (UDP, TCP, ICMP) began on the network associated with 192.168.5.28	Netflow
2016-08-25 19:52:15	Snort alert triggered saying that a DNS query was made directly to an external DNS server	Snort
2016-08-25 19:52	Computer queried external DNS server (8.8.8.8) requesting IP address for sra.com	Snort Alert DNS log
2016-08-25 19:52	Confirmed DNS queries made to 8.8.8.8 requesting the IP address for sra.com; answer returned was 163.252.95.35.	Netflow, DNS logs
	No traffic was directed to 163.252.95.35	Netflow
2016-08-25 19:56:27	Firefox (47) used to access the website at http://euspba.org	Squid – access.log, tshark
2016-08-25 19:56:31	Webpage http://euspba.org/thevoice.aspx was accessed via link on the eusba.org web page.	Squid – access.log, tshark
2016-08-25 19:56:38	Web page containing a PDF file was accessed via link on the http://euspba.org/thevoice.aspx web page.	Squid – access.log, Tshark, wireshark
2016-08-25 19:58:07	Last packet from 192.168.5.28 was seen	Netflow
2016-08-26 7:40	Computer not present on network; Noted by the absence of a DHCP lease renewal request.	DHCP log

In addition to being able to construct a timeline of the events, we obtained a copy of the PDF file from the network artifacts that was downloaded from the website. A copy was extracted from the full packet capture and compared to the original. The MD5 hash of each file matched, indicating the extract was an exact match of the file that was downloaded.

It should be noted that multiple artifacts were used to identify each event. These are important in that they corroborate the existence of the event given that multiple sources indicate it happened. Multiple sources of information are important in that some of the sources might not be available at the time when an analysis is performed. For

example, the analysis might be done after the retention time for a full packet capture has expired. Squid caches information for a specified period of time. Once that time has passed, the information might not be retrievable from Squid's cache.

5. Conclusion

This paper identified key artifacts that are important to support network forensics during incident response. It discussed the setup of a home lab architected to collect the artifacts using open source tools and validated the implementation of the architecture through a test scenario.

The test scenario was a simple test designed to perform some common activities on the network. The validation challenge presented was to see if, by using the network artifacts collected by the proposed architecture, an analyst could reconstruct the actions taken by the tester. The test was a success. It was possible to generate a timeline of the user's actions on the network as shown in Table 1.

The architecture used tools that are representative of tools that could be used for the collection of different types of artifacts. There are other tools that can serve the same purpose. For example SiLK, the System for Internet-Level Knowledge, is an open-source tool that provides similar functionality to nfdump.

More can be added to the architecture. Logging from the firewall could have been used in the analysis. Software for log aggregation could be added.

The architecture presented in this document provides a good starting point. It provides an environment in which to experiment and develop experience and skills without a big price tag. It also defines basic functionality that one might expect in a work environment.

Appendix A: Software Installation and Configuration

A.1 Firewall/Router

To setup a Linux server as a router IP Forwarding needs to be configured. This is configured by setting the following in the /etc/sysctl.conf file:

```
net.ipv4.ip_forward = 1
```

There are two options for firewalls on Centos 7 systems – firewalld and iptables. Firewalld is the newer of the two and the default. Iptables was chosen for this exercise. In order to use iptables, it must be installed and firewalld must be disabled. This is done using the commands:

```
yum install iptables-services
systemctl stop firewalld
systemctl disable firewalld
```

The key commands in a bash file that are needed to setup the routing using iptables with NAT are:

```
#!/bin/bash
INTERNET="enp2s0"
LAN="enp5s2"
iptables -t nat -A POSTROUTING -o $INTERNET -j MASQUERADE
iptables -A FORWARD -i $INTERNET -o $LAN -m state /
      --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i $LAN -o $INTERNET -j ACCEPT
```

Much more should be done to configure iptables, but that is beyond the scope of this paper. Some additional iptables commands are described in later sections of this paper where appropriate. Logging is not required, nor is it automatic when using iptables. Logging must be configured.

Iptables is established to run as a service using the command:

```
systemctl enable iptables
systemctl start iptables
```

A bash file, like that started above can be used to initialize the iptables rules. These rules can then be saved so that they persist beyond a reboot of the system using the command:

```
service iptables save
```

This will save the rules in the file /etc/sysconfig/iptables.

A.2 Time Services (NTP)

The firewall/router was established as the internal NTP server. NTP was installed using the command:

```
yum install ntp
```

The NTP server configuration file, /etc/ntp.conf, was modified to allow clients on the local network to synchronize their time with the time server using:

```
 # Hosts on local network are less restricted.
restrict 192.168.5.0 mask 255.255.255.0 nomodify notrap
```

Additionally, the NTP server was configured to synchronize itself with Internet-based time server with:

```
# Use public servers from the pool.ntp.org project.
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

The following rules were added to iptables to allow the NTP server to synchronize their time with the Internet-based time servers and to allow internal servers to synchronize with it.

```
iptables -A OUTPUT -o $INTERNET -p udp --dport 123 -j ACCEPT
iptables -A INPUT -i $INTERNET -p udp --sport 123 -j ACCEPT
iptables -A INPUT -i $LAN -p udp --dport 123 -j ACCEPT
iptables -A OUTPUT -o $LAN -p udp --sport 123 -j ACCEPT
```

For the NTP clients on the local network, the server command needs to be modified to add the internal NTP server, firefly, as shown below and the public servers need to be commented out.

```
# Sync with internal ntp time server
server firefly.fraser.local iburst

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#server 0.centos.pool.ntp.org iburst
```

```
#server 1.centos.pool.ntp.org iburst
#server 2.centos.pool.ntp.org iburst
#server 3.centos.pool.ntp.org iburst
```

The systemctl command can be used to manage the NTP server and the NTP clients.

A.3 Dynamic Host Configuration Protocol (DHCP)

The firewall/router was established as the internal DHCP server. DHCP was installed using the command:

```
yum install dhcp
```

The following is the DHCP configuration (/etc/dhcp/dhcpd.conf) for the lab.

```
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.example
#   see dhcpd.conf(5) man page
#
ddns-update-style none;

option domain-name-servers 192.168.5.1;
option domain-name "fraser.local";
default-lease-time 86400;    # time in seconds - 1 day
max-lease-time 259200;       # time in seconds - 3 days
authoritative;
log-facility local7;

subnet 192.168.5.0 netmask 255.255.255.0 {
  range 192.168.5.26 192.168.5.99;
  option routers 192.168.5.1;
  option broadcast-address 192.168.5.255;
}
```

Since the DHCP server is located on the firewall system, rules need to be established to allow the systems on the internal network to communicate with it.

```
iptables -A INPUT -i $LAN  -p udp --sport 67 -j ACCEPT
iptables -A INPUT -i $LAN  -p udp --sport 68 -j ACCEPT
iptables -A OUTPUT -o $LAN -p udp --dport 67 -j ACCEPT
iptables -A OUTPUT -o $LAN -p udp --dport 68 -j ACCEPT
```

DHCP logs to /var/log/messages.

A.4 Domain Name Server (DNS)

The firewall/router was established as the internal DNS server. DNS was installed using the command:

```
yum install bind bind-utils
```

There are several configuration files that need to be setup to make DNS work. These include the /etc/named.conf, /var/named/forward.zone, and /var/named/reverse.zone.

Since the internal DNS server is located on the firewall system, rules need to be established to allow the systems on the internal network to communicate with it and for it to communicate with an upstream DNS to resolve external addresses. The iptables rules are as follows:

```
iptables -A OUTPUT -o $INTERNET -p udp -d 192.168.1.1/32 --dport 53 -j LOG --
log-prefix "iptables: "
iptables -A INPUT -i  $INTERNET -p udp -s 192.168.1.1/32 --sport 53 -j LOG --
log-prefix "iptables: "

iptables -A OUTPUT -o $INTERNET -p udp -d 192.168.1.1/32 --dport 53 -j ACCEPT
iptables -A INPUT -i  $INTERNET -p udp -s 192.168.1.1/32 --sport 53 -j ACCEPT
iptables -A OUTPUT -o $INTERNET -p tcp -d 192.168.1.1/32 --dport 53 -j ACCEPT
iptables -A INPUT -i $INTERNET -p tcp -s 192.168.1.1/32 --sport 53 -j ACCEPT
```

A.5 Full Packet Capture

Full packet captures are valuable in that they provide a complete picture of what crossed the network. One of the most popular applications used for capturing packets is tcpdump (Banks, 2013). Tcpdump was set up as a service on the monitoring server.

There are a few considerations that must be taken into account when setting up tcpdump as a service. Root access is generally required to capture network traffic, but it is not desirable to run the service as root. Systemd is the service manager for Centos Linux. It initiates processes initially as root and then shifts to the user tcpdump. The capture files are stored owned by the user tcpdump.

Running services within Systemd is done through the use of unit files. Here is a Systemd unit file to run tcpdump as a service. This configuration executes a script,

/usr/local/scripts/tcpdump.sh, with a user id of tcpdump and group of tcpdump. Restart is configured and it uses the process id (pid) that is stored in /var/log/tcpdump/tcpdump.pid.

```
[Unit]
Description=tcpdump capture daemon
After=network.target

[Service]
Type=forking
ExecStart=/usr/local/scripts/tcpdump.sh
PIDFile=/var/log/tcpdump/tcpdump.pid
Restart=always
User=tcpdump
Group=tcpdump

[Install]
WantedBy=multi-user.target
Alias=tcpdump.service
```

The following is the script that the unit file executes.

```
#!/bin/bash
IFACE="enp1s0"
DATADIR="/var/log/tcpdump"
cd $DATADIR

# start process in backgroud
/usr/sbin/tcpdump -s0 -nn -i $IFACE -G3600 -w "%Y%m%d_%H%M.pcap" &
sleep 5

# get pid
echo $! > /var/log/tcpdump/tcpdump.pid
```

Several other system configurations must be made for this to work. The /var/log/tcpdump directory must be owned by the tcpdump user and group to allow the process to write to it. In order to permit a non-root user to capture traffic using tcpdump, the setcap utility must be run.

```
$ setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
```

This setting can be verified using the command:

```
$ getcap /usr/sbin/tcpdump
/usr/sbin/tcpdump = cap_net_admin,cap_net_raw+eip
```

The service can be managed – started, enabled at boot, stopped, etc. -- using the systemctl command.

A.6 DNS Logging

PassiveDNS was installed from source code according to the instructions provided on the PassiveDNS github site (https://github.com/gamelinux/passivedns/blob/master/doc/INSTALL).

```
$ yum groupinstall "Development tools"
$ cd passivedns/
$ autoreconf --install
$ ./configure
$ make
```

To configure PassiveDNS as a systemd service, the same process as used for tcpdump was used. The /etc/systemd/system/passivedns.service unit file is:

```
[Unit]
Description=passivedns logging daemon
After=network.target

[Service]
Type=forking
ExecStart=/usr/local/scripts/passivedns.sh
PIDFile=/var/log/passivedns/passivedns.pid
Restart=always
User=tcpdump
Group=tcpdump

[Install]
WantedBy=multi-user.target
Alias=passivedns.service
```

The following script, /usr/local/scripts/passivedns.sh, is called from the systemd unit file to start the service.

```
#!/bin/bash

IFACE="enp1s0"
DATADIR="/var/log/passivedns"
FILENAME=$(date +"%Y%m%d_%H%M.log")
cd $DATADIR

# start process in backgroud
```

```
/usr/local/bin/passivedns -i $IFACE -l $FILENAME -L
/var/log/passivedns/passivedns.log -p /var/log/passivedns/passivedns.pid -u
tcpdump -g tcpdump -D &
```

Since PassiveDNS reads the traffic from the network interface using libpcap, the setcap command must be run to grant the executable the capability to capture traffic as a non-root user.

```
$ setcap cap_net_raw,cap_net_admin=eip /usr/local/bin/passivedns
```

The log file directory, /var/log/passivedns, needs to be owned by the user tcpdump and the group tcpdump.

A.7 Netflow

Nfpcapd was installed from source code according to the instructions provided on the nfdump github site (Haag, 2015). The option for the ./configure command –enable-nfpcapd was used to allow nfpcapd to be used to read pcap files created by tcpdump.

Configuring netflow packet capture as a systemd service was done similar to what was done for tcpdump. Here is the /etc/system/system/nfpcapd.service unit file:

```
[Unit]
Description=netflow logging daemon
After=network.target

[Service]
Type=forking
ExecStart=/usr/local/scripts/nfpcapd.sh
PIDFile=/var/log/nfdump/nfpcapd.pid
Restart=always
User=tcpdump
Group=tcpdump

[Install]
WantedBy=multi-user.target
Alias=nfpcapd.service
```

The following is the script, /usr/local/scripts/nfpcapd.sh, that the nfpcapd.service file calls:

```
#!/bin/bash
```

```
IFACE="enp1s0"
DATADIR="/var/log/nfdump"
cd $DATADIR

# start process in backgroud
/usr/local/bin/nfpcapd -i $IFACE -l /var/log/nfdump -P
/var/log/nfdump/nfpcapd.pid -t 300 -D
```

Since nfpcapd reads traffic from the network interface using libpcap, we need to use the setcap command to grant the executable the capability to capture traffic as a non-root user.

```
$ setcap cap_net_raw,cap_net_admin=eip /usr/local/bin/nfpcapd
```

The log file directory, /var/log/nfdump, needs to be owned by the user tcpdump and the group tcpdump.

A.8 Intrusion Detection System (Snort)

Snort was installed using the instructions by William Parker (Parker, 2015). The version of Snort was 2.9.8.0, the version of DAQ was 2.0.6, and the version of Snort rules was 2980. Where the installation deviated from Parker's instructions was with the startup script. In order to use systemd services, the system service file, snort.service, listed below was used instead.

```
[Unit]
Description=Snort IDS Daemon
After=syslog.target network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i
enp1s0

[Install]
WantedBy=multi-user.target
```

A.9 Arpwatch

Arpwatch was installed using the command:

```
sudo yum install arpwatch
```

Once Arpwatch was installed, the environment file, /etc/sysconfig/arpwatch, was modified to add the interface to the options using "`-i enp6s0`". It appears that the

interface must have an IP address to work. The installation includes the systemd unit file, aprwatch.service, so the systemctl commands can be used to start, stop, enable, and check the status of the process. Arpwatch logs to /var/log/messages.

A.10 Proxy Server (Squid)

Squid ProxyServer, version 3.3.8, was installed using the command:

```
sudo yum install squid
```

Once the software was installed several configuration changes needed to be made in the /etc/squid/squid.conf file. The configuration parameter `pid_filename /var/log/squid/squid.pid` needed to be added to correct a permissions problem. The line beginning with cache_dir needed to be uncommented to enable caching. The word transparent needed to be added to the end of the line beginning with http_port to enable transparent proxying.

In order to make the squid server work the following lines needed to be added to the iptables configuration:

```
iptables -t nat -A PREROUTING -i $LAN -p tcp --dport 80 /
      -j REDIRECT --to-port 3128
iptables -A INPUT -i $LAN -p tcp --dport 3128 -m state /
      --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o $LAN -p tcp --sport 3128 -m state /
      --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o $INTERNET -p tcp --dport 80 -m state /
      --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i $INTERNET -p tcp --sport 80 -m state /
      --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o $LAN -p tcp --sport 80 -m state /
      --state ESTABLISHED,RELATED -j ACCEPT
```

The installation includes the systemd unit file, squid.service, so the systemctl commands can be used to start, stop, enable, and check the status of the process.

References

Banks, Derek. (2013). Custom Full Packet Capture System. Retrieved April 18, 2016, from https://www.sans.org/reading-room/whitepapers/logging/custom-full-packet-capture-system-34177.

Davidoff, Sherri and Ham, Jonathan. (2012). Network Forensics: Tracking Hackers through Cyberspace. Prentice Hall: Upper Saddle River, NJ.

Fjellskål, Edward Bjarte. (August 2015). PassiveDNS Readme. Retrieved April 18, 2016, from https://github.com/gamelinux/passivedns.

Fjellskål, Edward Bjarte. (December 2015). PassiveDNS Install. Retrieved April 18, 2016 from https://github.com/gamelinux/passivedns/blob/master/doc/INSTALL

Haag, Peter. (2015). Nfdump Readme. Retrieved April 18, 2016, from https://github.com/phaag/nfdump.

Hagen, Phil (2015a). Advanced Network Forensics and Analysis: Logging and OPSEC, and Footprint. The SANS Institute: Bethesda, MD.

Hagen, Phil. (2015b). Advanced Network Forensics and Analysis: Netflow Analysis, Commercial Tools, and Visualization. The SANS Institute: Bethesda, MD.

Hagen, Phil. (2015c). Advanced Network Forensics and Analysis: Network Protocols and Wireless Investigations. The SANS Institute: Bethesda, MD.

Hagen, Phil (2015d). Advanced Network Forensics and Analysis: Off the Disk and Onto the Wire. The SANS Institute: Bethesda, MD.

Mandiant. (February, 2016). M-Trends 2016. FireEye, Inc.: Milpitas, CA. Retrieved April 15, 2016 from https://www2.fireeye.com/M-Trends-2016.html.

Parker, William. (December 24, 2015). Getting Snort working in Centos 6.x/7.x and Virtual Box 5.x.x. Retrieved July 1, 2016, from https://www.snort.org/documents.

Skoudis, Ed, Strand, John, and SANS. (2014). Incident Handling Step by Step and Computer Crime Investigation. The SANS Institute: Bethesda, MD.

Squid Wiki. (November 18, 2015). Squid Log Files. Retrieved August 25, 2016, from http://wiki.squid-cache.org/SquidFaq/SquidLogs#Squid_result_codes

Gordon Fraser, Gordon.fraser@ctipc.com

Last Updated: November 17th, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS San Francisco Winter 2017	San Francisco, CAUS	Nov 27, 2017 - Dec 02, 2017	Live Event
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZUS	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Khobar 2017	Khobar, SA	Dec 02, 2017 - Dec 07, 2017	Live Event
SANS Munich December 2017	Munich, DE	Dec 04, 2017 - Dec 09, 2017	Live Event
European Security Awareness Summit & Training 2017	London, GB	Dec 04, 2017 - Dec 07, 2017	Live Event
SANS Austin Winter 2017	Austin, TXUS	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Frankfurt 2017	Frankfurt, DE	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS Bangalore 2017	Bangalore, IN	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DCUS	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS SEC460: Enterprise Threat Beta	San Diego, CAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
Northern VA Winter - Reston 2018	Reston, VAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SEC599: Defeat Advanced Adversaries	San Francisco, CAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SANS Amsterdam January 2018	Amsterdam, NL	Jan 15, 2018 - Jan 20, 2018	Live Event
SANS Dubai 2018	Dubai, AE	Jan 27, 2018 - Feb 01, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NVUS	Jan 28, 2018 - Feb 02, 2018	Live Event
Cyber Threat Intelligence Summit & Training 2018	Bethesda, MDUS	Jan 29, 2018 - Feb 05, 2018	Live Event
SANS Miami 2018	Miami, FLUS	Jan 29, 2018 - Feb 03, 2018	Live Event
SANS London February 2018	London, GB	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Scottsdale 2018	Scottsdale, AZUS	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Southern California- Anaheim 2018	Anaheim, CAUS	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS Secure India 2018	Bangalore, IN	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS London November 2017	OnlineGB	Nov 27, 2017 - Dec 02, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced