Global Information Assurance Certification Paper

Integration of Network Conversation Metadata with Asset and Configuration Management Databases

GIAC (GCIA) Gold Certification

Author: William Yeatman, wmyeatman@gmail.com
Advisor: Barbara Filkins
Accepted: May 16, 2015

Abstract

As an alternative the loss of access to plaintext IP payloads in an increasingly encrypted and privacy conscious world, network layer security analysis requires a shift of attention to examination and characterization of the packet and network conversation meta-information derived from packet header information. These characteristics can be incorporated into and treated as an integral part of asset and configuration management baselines. Changes detected in the expected endpoints, frequency, duration, and packet sizes can be flagged for review and subsequent response or adjustment to the baseline.

1. Introduction

The use of encryption to protect the confidentiality of network communications is on the rise. According to Sandvine's 2014 Global Internet Phenomena Report, and as reported by Wired Magazine, the volume of encrypted online communications has nearly doubled in North America, tripled in Europe, and more than quadrupled in Latin America since the 2013 Snowden revelations (Finley, 2014).

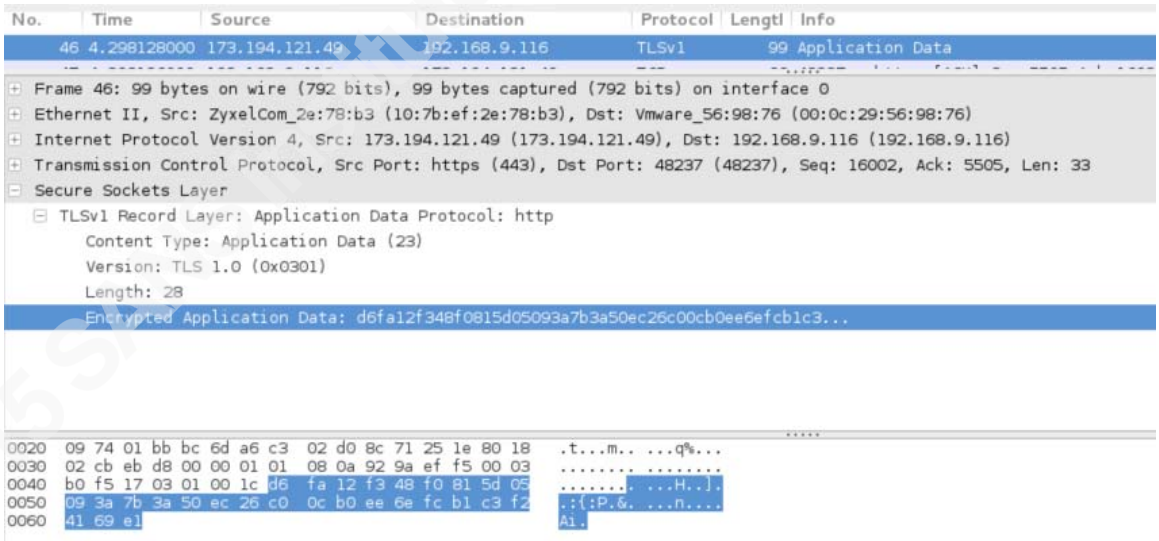
Consumer end users are not the only ones to see a rise in the use of network level encryption. Enterprise encryption strategies show a trend toward encryption across both external and internal networks. In the Ponemon Institute's 2014 *Global Encryption Trends Study*, 35% of respondents indicated their organization had extensively deployed solutions for the encryption of external public networks, enterprise wide, with 47% of respondents indicating they had at least started with partial or "point" deployments (Ponemon, 2014). For encryption of internal network traffic, 32% of the respondents indicated the extensive use of encryption technologies enterprise-wide and 46% indicated some level of progress with partial deployments.

In the wake of high profile cyber security breaches and allegations of government snooping into Internet traffic, proliferation of interest in the use of encryption for the protection of network communications should come as no surprise. After all, for security professionals, confidentiality is a core tenet of a good information security program. But the good guys are not the only ones using encryption to try to protect privacy of communications and foil cyber attacks. Malware Command and Control (C2) traffic is usually encrypted nowadays and deprecated is the use of plaint-text C2 avenues such as Internet Relay Chat (IRC) (Grosfelt, 2014).

Contrary to the notion of protecting the confidentiality of communication to strictly the sender and receiver, the security analyst's job is (was) often easier and more colorful when the analyzed traffic is (was) plaintext. What commands was that malware trying to execute and what other hosts may have been implicated? Easy! Pull the IRC related network packets, fire up Wireshark or your favorite protocol decoder, and pick through the strings. Obviously, plaintext payloads can be accessed, parsed, readily

understood, and often make for much more insightful and interesting analysis by the security professional. An interesting question is raised. What happens when all (or at least the majority of) network traffic is encrypted and the security analyst does not have the decryption keys (which is usually the case when it comes to modern malware)? And even when the analyst has access to the decrypted network stream, as in the case of SSL proxies, there are times where the use of such a proxy is detectable by the end server and terminated in the interest of absolute confidentiality and preventing a man in the middle MITM attack.

So, of what use to us is the network security analyst in a world where all packet payloads are encrypted? There are many answers, of course, and the good news is that network monitoring is not dead just because the payloads and application content are encrypted. But first, it is good to clarify what is meant by network conversation metadata. For the remainder of this paper, network conversation metadata refers to the plaintext header data and statistical information that can be derived from Internet Protocol (IP) version 4 or 6 packets. The following packet capture is a basic illustration of an IPv4 packet that has its encrypted payload:



Note that the highlighted portion of the packet show that that payload is encrypted, but the IP and Transmission Control Protocol headers are all still plaintext. There is much to be gleaned from the IP and TCP headers – note above that we have access to basic information such as the source and destination addresses, port numbers, TCP sequence

numbers, and the like. Inherent also is information such as packet lengths, payload lengths (whether encrypted or not), and other data that could be used to extrapolate and characterize the conversations between networked peers. We are advised: "turning to session data or statistics on the sorts of ports and addresses is a better way to identify suspicious activity" (Bejtlich, 2005). This approach has found practical application in the identification of compromised healthcare organizations by analyzing just source/destination IP addresses and geo-location information over a period of one year (Filkins, 2014).

It stands to reason that an asset-centric approach to monitoring may be helpful. There would seem to be utility in capturing information about network conversations from the wire and integrating the characteristics of the conversations into asset/configuration management baselines. At a high level, this is akin to peer whitelisting, but at the asset/configuration management level. In the asset/config management system, we consciously annotate that host X was observed communicating with host Y or host Z using some protocol (e.g., TCP) on some port (443, for example). Other potentially interesting metadata such as conversation duration, packet counts, and byte sizes could also be recorded and integrated as part of the baseline and configuration management for host X.

While establishment of system profiles relative to event/incident response efficiencies is not an entirely new concept (Karwaski, 2009), and approaches to monitoring network behavior (i.e., tools for analysis of NetFlow records, Silk) have been available for many years, the notion is advanced when we consider actually integrating the network conversation characteristics as part of the baseline configuration for an asset.

Again, the idea is that most packet payloads will presumably, at some future point, be unavailable for analysis due to increasingly prevalent use of transmission encryption. Network layer security analysis thus continues shifting focus toward the monitoring of the meta- and statistical information of network conversations during analysis. The remainder of this paper discusses the concept of integrating this network conversation meta-information into an existing asset/configuration management system. Potential benefits and challenges are shared. Finally, a working example is presented to

offer a tangible view of what an approach of this nature might offer in the protection of critical systems.

2. Handling Network Conversation Information as Asset/Configuration Item

2.1. A Proposal

SANS Critical Security Controls (CSC) 1-1 and 2-2 recommend the deployment of an automated asset inventory discovery tool and its use in building an asset inventory of systems and software in use at an organization (SANS, 2014). CSC 1-4 and 2-4 then indicate that the following attributes should be maintained in the inventory:

- Network address
- Machine name
- Purpose of system
- Asset owner
- Department associated with the device
- Operating system w/ version and patch level
- Installed applications w/ version and patch level

The addition of network conversation meta-information to this asset inventory, or a configuration management database tied directly to it, would support an asset-based, continuous view of the conversations occurring on the network.

What is interesting is the idea of using existing security analyst and network capture tools to augment information in the asset/configuration management databases. Such an approach provides visibility to system administrators and owners as to what their systems are talking to, and which systems are talking to them. Integration of this information with the asset/configuration management databases could facilitate review of these conversations and provide an opportunity for formal approval as part of a baseline for any particular set of systems. In the end, this helps answer the question "Do we know

what is connecting to and running (or trying to run) on our systems and networks?" (SANS, 2014).

Ultimately, the revised asset or configuration management database might appear as follows:

- Network address
- Machine name
- Purpose of system
- Asset owner
- Department associated with the device
- Operating system w/ version and patch level
- Installed applications w/ version and patch level
- **NEW: Approved Network Conversation Peers**

As network layer intelligence is obtained from the wire, it can then be fed into the asset/configuration management database where automatic comparisons can be made to identify anomalies against the approved list of hosts authorized to communicate with the asset in question. Ideally, the asset/configuration management interface would allow the system administrator or owner to indicate that a newly identified connection is authorized - or to potentially trigger the incident response process if it was not expected!

Ideally, the feeding of network layer intelligence into the asset/configuration management system would be accompanied by correlation with other information sources that aid the system administrator/owner in making a determination as to whether or not the peers in the conversation are authorized to be communicating with one another. For example, if the peer on either side of the observed connection is not already in the asset/configuration database, then DNS, geolocation data, threat intelligence feeds and other block list information can be queried and presented to the system administrator/owner. Putting this information, automatically at the fingertips of the system administrator/owner reduces the amount of additional research that would need to be performed in discerning whether any given peer-to-peer conversation is expected.

William Yeatman, wmyeatman@gmail.com

Additional consideration would need to be given to the amount and granularity of information that should be included.

2.1.1. Challenges

It is worth mentioning, upfront, the potential drawbacks to this asset-centric approach. The first is, as any good security analyst knows, the potential to produce large volumes of data that no administrator or system owner would practically be able to sift through. This may be the case particularly with end user workstations and mobile devices that have unfettered Internet connectivity. Overwhelming system administrators with more data that leads to more work is not the goal of this exercise, so any attempt to incorporate network traffic observations with the asset/configuration management system needs to be done in a manner that reduces the amount of work that needs to be done in researching and identifying peers (see discussion in Section 2.1 regarding automatically integrating geo-location and other intelligence sources).

A second challenge is that the integration of network conversational data with the asset/configuration management database will be incomplete if the network is not properly equipped to capture all conversations. For example, in a network with only one packet capturing device sitting on the external side of their Internet facing firewall (i.e., an umbrella IDS/sensor), it is possible that only packets that have had Network Address Translation applied (e.g., packets that have been NATed), it would be difficult to integrate with the approach described here since the original source IP would not be available. This also becomes apparent when the topic of off-network devices and intra-segment communications comes up. In the case of the former, it may prove difficult to capture all communications from mobile devices that switch between, for example, the monitored corporate wifi network and a wireless 4G carrier network. In the case of the latter, it is the experience of the author that few organizations are willing to invest in sensor infrastructure capable of capturing all network traffic within a single segment or Virtual Local Area Network (VLAN).

It is worth mentioning that a host-based approach could be pursued wherein the desired network conversation information is obtained directly from the host system and fed into the asset/configuration management database. While this may well prove to be a

robust and informative source of network conversation information, and necessary complement to any network layer monitoring initiative, further discussion is outside the scope of this paper.

Ultimately, the integration of network layer conversation metadata with asset/configuration management systems may be better suited to environments that have relatively predictable sets of network communication peers. This could include internal database/application/middleware servers, supervisory control and data acquisition (SCADA) components, point of sale (POS) terminals, and other purpose built systems. Taking the example of SCADA components to illustrate further, these are systems that are typically recommended for placement within an air-gapped network that is physically separate from any other network (Scott, 2014). Similarly, the Payment Card Industry Data Security Standard 3.0 (PCI DSS 3.0) mandates the separation of systems that store, process, or transmit cardholder information and the implementation of a "deny-by-default" access posture. These types of environments, while perhaps not completely immune to unauthorized access or compromise, do lend themselves to the formulation of predictable sets of communication peers.

2.1.2. Benefits

Making network conversation meta-information a discrete configuration item in system inventories and configuration management solutions, and putting intelligence information about the peers involved readily at the fingertips of system administrators/owners potentially helps to identify unauthorized or unusual communications. This fosters an attitude of keeping the shop clean and knowing what's going on across the network, and the approach could be very useful for organizations that have minimal staff dedicated to security monitoring.

Second, the opportunity to validate preventive controls is provided. Firewalls can be misconfigured, can fail open, and malware solutions do not catch everything. Inclusion of host-to-host conversations observed on the network and integrated with the asset/configuration management system provides a detective control for assuring other controls are properly configured and operating as expected.

Finally, if implemented properly, including network conversation information in the asset/configuration management system could allow for a historical/longitudinal view regarding the conversations in which any particular asset has participated. With the exception of aforementioned mobile and end user systems, and publicly accessible services, the number of peer systems with which a critical asset must communicate should be relatively easy to track via the asset/configuration management system.

2.2. Working Example

In order to help visualize how the network connection metadata may appear within the context of an asset/configuration management system, a working example is provided. There are a wide variety of tools and approaches that can be used to accomplish the task of capturing and integrating network conversation information with asset and configuration management databases. The proof of concept shared here is based entirely on open source tools and software.

Further exploration of ways to best achieve integration with any particular asset/configuration management system is left as future work, and an exercise for the reader.

2.2.1. Configuration Overview and Tools Utilized

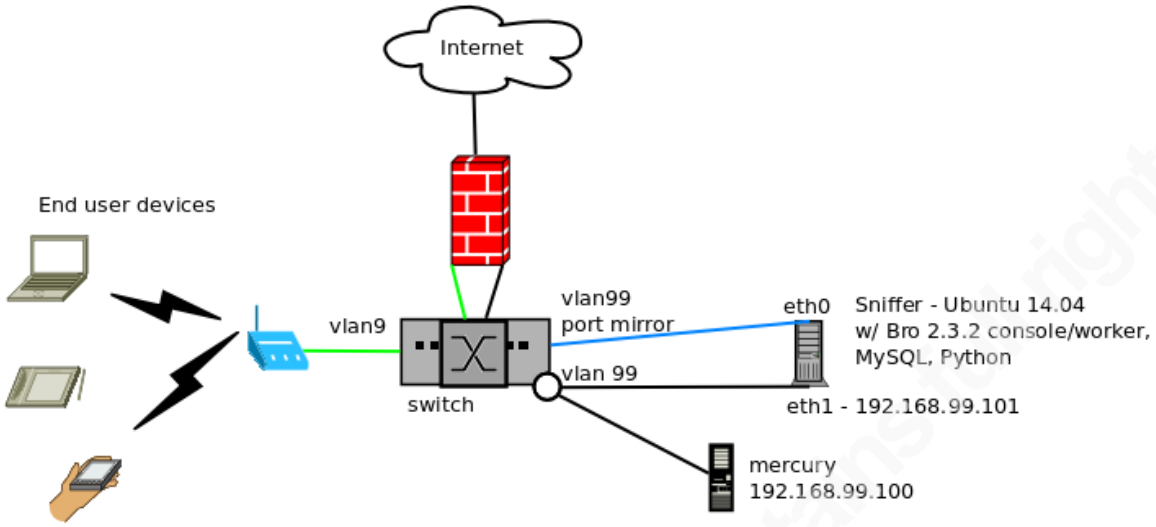
The following tools were used to create the working example:

- Bro 2.3.2 running on Ubuntu 14.04.1 LTS was used for capturing packets and connection information; Bro was selected because its connection log provides the fastest way to glean high level information about the nature of network connections observed on the network. This includes source and destination IP address (both IP v4 and v6), protocol, port, bytes/packets exchanged, connection session duration, and other interesting meta-information about the conversation; additional information about Bro can be obtained from www.bro.org;
- MySQL 5.6.19 is used for storing the connection information captured by Bro. While Bro does a great job of capturing the important connection information, it does not support logging to a database and typically stores

its logs across multiple files that get compressed and archived on an hourly basis by default. When taking an asset based view of network conversation information, storing events across multiple compressed archives could become unwieldy. Storing the connection event information in a single, easily queried location facilitates the ability to achieve a broader longitudinal view and seems more appropriate for gaining an asset based perspective;

- Python 2.7.6 is used for reading the Bro connection logs and inserting data about network talkers into the MySQL database for ease of query. Several additional python modules are installed to facilitate IP address conversions, connections to the mysql database (see appendix for code listing, which includes the names of the imported modules);
- MaxMind Insight Subscription (50,000 query license) is used to obtain geographical and other information about each public IP address observed in the packet captures. As the asset/configuration centric picture is built around the conversations occurring on the network, it becomes important to quickly be able to identify the "traits" of an IP address, particularly those that are external to the organization. Having the Autonomous System Number (ASN), Internet Service Provider (ISP), domain, geolocation and other information about any given IP address at the fingertips of someone reviewing connection information is key for efficiency when analyzing and reviewing.

The following network was used in creating this working example:



Note that "mercury" is the name of a test server that has tightly controlled egress access at the firewall and will serve as the main asset that will be included and examined in the example. Bro is started via broctl on the sniffer server and captures all VLAN 99 traffic on its eth0 interface (including mercury's and its own traffic), capturing all connection information in the directory /usr/local/bro/logs/[yyyy-mm-dd]. Thus, it records connections made by/to 192.168.99.100 and 192.168.99.101. The switch in the above diagram is a D-Link DGS-1100-08 switch with port mirroring capabilities.

After installation of the toolset and successful configuration of the above, traffic is captured for 36 hours.

2.2.2. Processing Bro's Connection Data

Once Bro produces sufficient data, a python script called broconns.py is run to process the archived Bro connection files. A listing of this very crude but effective code is provided in the appendix.

To summarize its functionality, broconns.py is pointed to the location of the bro log files. It recursively descends into each bro log subdirectory, finding and processing the compressed connection logs. For each log file that it processes, the script extracts the relevant information for each network connection event and inserts it into the backend mysql database. The script also performs a lookup against the geolocation service to glean intelligence about the IP addresses it comes across and adds that information to the

database for later reference. Since Bro can produce many connection log files over time, broconns.py keeps track of the files that it has processed in order to prevent duplicate processing.

2.2.3. Providing the Asset/Configuration Management Perspective

After allowing bro to collect sample data and importing it into the mysql database, the database can be queried to illustrate how the connectivity information might appear in the context of an asset/configuration management system. Let's start by looking at host mercury, as shown in Figure 1. above. Mercury's IP address is 192.168.99.100 and we are using it as a secure file server used by the Information Security department. A mock asset/configuration management database was created to reflect this:

```
mysql> select sip4, machine_name, purpose, owner, dept, os from cmdb where sip4='192.168.99.100';
+---------------+--------------+-------------------+--------------+---------------------+-----------------+
| sip4          | machine_name | purpose           | owner        | dept                | os              |
+---------------+--------------+-------------------+--------------+---------------------+-----------------+
| 192.168.99.100 | mercury      | Secure File Server | Alice Roberts | Information Security | Ubuntu 14.04 LTS |
+---------------+--------------+-------------------+--------------+---------------------+-----------------+
1 row in set (0.00 sec)
```

Next, let's take a look at the hosts with whom mercury has been communicating. The following screenshots are snippets of the results from running this SQL query:

```
mysql> select dip4 as 'Dst IP', prot, dport, max(cast(duration as
unsigned)) as 'Max Duration', min(cast(duration as unsigned)) as 'Min
Duration', max(cast(orig_bytes as unsigned)+cast(resp_bytes as
unsigned)) as 'Max Bytes', min(cast(orig_bytes as
unsigned)+cast(resp_bytes as unsigned)) as 'Min Bytes', geoip_country
as 'dst_country', geoip_city as 'dst_city', geoip_org as 'dst_org',
geoip_dom as 'dst_domain', status from conn4 join ip4_intel on dip4=ip4
where sip4='192.168.99.100' and dip4 NOT LIKE '192.168%' and duration
```

```
!= '-' group by sip4, dip4, prot, dport;
```

Dst IP	prot	dport	Max Duration	Min Duration	Max Bytes	Min Bytes
108.168.255.243	tcp	443	0	0	750	747
162.159.242.165	tcp	443	121	0	4970	199
162.222.40.154	tcp	443	2700	300	2516565	32
216.17.8.47	tcp	443	2701	299	1239	32
54.230.16.20	tcp	443	3	3	0	0
54.230.16.242	tcp	443	3	3	0	0
54.230.17.182	tcp	443	3	3	0	0
54.230.19.186	tcp	443	3	3	0	0
54.230.19.214	tcp	443	3	3	0	0
54.240.160.139	tcp	443	3	3	0	0
67.215.92.215	tcp	80	0	0	91	91
74.125.29.95	tcp	443	3	3	0	0
82.98.134.233	icmp	0	2	2	168	168

dst_country	dst_city	dst_org	dst_domain	status
United States	Dallas	MaxMind LLC	softlayer.com	New connection, requires review
United States	San Francisco	CloudFlare	None	New connection, requires review
United States	Minneapolis	Code 42 Software	crashplan.com	Approved by Alice, this is a fil
United States	Minneapolis	Code 42 Software	crashplan.com	Approved by Alice, this is a fil
United States	Seattle	Amazon.com	None	New connection, requires review
United States	Seattle	Amazon.com	None	New connection, requires review
United States	Seattle	Amazon.com	None	New connection, requires review
United States	Seattle	Amazon.com	None	New connection, requires review
United States	Seattle	Amazon.com	None	New connection, requires review
United States	Seattle	Amazon.com	None	New connection, requires review
United States	San Francisco	OpenDNS, LLC	opendns.com	New connection, requires review
United States	Mountain View	Google	1e100.net	New connection, requires review
Spain	NULL	DinaHosting S.L.	dinaserver.com	New connection, requires review

As shown above, the destinations with which host mercury has been communicating are listed in the query results, along with the destination protocol and port. Also included is information about each destination and statistical data about the session observed. The country, city, organization, and domain corresponding to each destination are all listed. This is all very important information to a system owner or administrator who is reviewing and acknowledging the legitimacy of each connection. All too often, this information is not directly at the fingertips of system administrators and owners, thus leading to inefficient and ad-hoc information gathering about peer communications. And while the example above is a rough sketch with simple command line queries, it demonstrates the possibility of providing a more complete picture regarding the network conversation activity for any given asset on the network.

Perhaps the above example includes too much information. No worry, let's roll it up and just show the destination organization and corresponding autonomous system

numbers of the hosts with which mercury (192.168.99.100) has initiated communication:

```
mysql> select geoip_org as 'Dst Org', geoip_asn_num, prot, dport, status from conn4 join ip4_intel on dip4=ip4
where sip4='192.168.99.100' and dip4 NOT LIKE '192.168%' and duration != '-' group by sip4, prot, dport, geoip_
org, geoip_asn_num;
+------------------------+--------------+------+-------+-----------------------------------------------------+
| Dst Org                | geoip_asn_num | prot | dport | status                                              |
+------------------------+--------------+------+-------+-----------------------------------------------------+
| Canonical Ltd          | 41231        | icmp |     0 | Approved by Alice, used for pulling OS Updates      |
| DinaHosting S.L.       | 42612        | icmp |     0 | New connection, requires review                     |
| Canonical Ltd          | 41231        | tcp  |    80 | Approved by Alice, used for pulling OS Updates      |
| OpenDNS, LLC           | 36692        | tcp  |    80 | New connection, requires review                     |
| Abovenet Communications | 17025       | tcp  |   443 | New connection, requires review                     |
| Amazon.com             | 16509        | tcp  |   443 | New connection, requires review                     |
| Canonical Ltd          | 41231        | tcp  |   443 | Approved by Alice, used for pulling OS Updates      |
| CloudFlare             | 13335        | tcp  |   443 | New connection, requires review                     |
| Code 42 Software       | 62715        | tcp  |   443 | Approved by Alice, this is a file backup destinati  |
| Google                 | 15169        | tcp  |   443 | New connection, requires review                     |
| MaxMind LLC            | 36351        | tcp  |   443 | New connection, requires review                     |
+------------------------+--------------+------+-------+-----------------------------------------------------+
11 rows in set (0.02 sec)
```

With Bro's connection data being stored in a database and integrated with the asset/configuration management database, so much more is possible. As a system administrator or owner reviewing this connection information and updating the status to flag if these are legitimate and expected connections, what if she wants to know more about the connections to opendns.com? This looks strange since this is our secure file server and all Domain Name Service (DNS) queries should be made against the internal DNS server. A good place to start is by checking when this was first seen on the network, and the last time it was observed:

```
mysql> select dip4, prot, dport, geoip_org, geoip_country, count(*) as '# Sessions', from_unixtime(min(ts)) as 'First Seen',
 from_unixtime(max(ts)) as 'Last Seen' from conn4 join ip4_intel on ip4 = conn4.dip4 where dip4 = '67.215.92.215';
+---------------+------+-------+-------------+--------------+------------+---------------------+---------------------+
| dip4          | prot | dport | geoip_org   | geoip_country | # Sessions | First Seen          | Last Seen           |
+---------------+------+-------+-------------+--------------+------------+---------------------+---------------------+
| 67.215.92.215 | tcp  |    80 | OpenDNS, LLC | United States |        135 | 2015-04-11 20:47:27 | 2015-04-12 15:58:23 |
+---------------+------+-------+-------------+--------------+------------+---------------------+---------------------+
1 row in set (0.01 sec)
```

Interestingly, this is not a DNS query, but an attempt to access a web server at OpenDNS. Perhaps its time for Alice to log in to the server and investigate why it is attempting to connect to OpenDNS.

This type of information is all readily available, starting with an asset-centric view and then expanding and exploring from there. Connections can subsequently be marked as authorized/verified.

William Yeatman, wmyeatman@gmail.com

Finally, IPv6 would need to be supported. In the working example, these connections are captured and reviewed in the same manner as demonstrated above for IPv4 connections:

```
mysql> select sip6, dip6, prot, dport, max(cast(orig_bytes as unsigned)+cast(res
p_bytes as unsigned)) as 'Max Bytes' from conn6 where sip6='fe80::b44b:2aef:e6f6
:c6ce' group by sip6, dip6, prot, dport;
+--------------------------+------------------+------+-------+-----------+
| sip6                     | dip6             | prot | dport | Max Bytes |
+--------------------------+------------------+------+-------+-----------+
| fe80::b44b:2aef:e6f6:c6ce | ff02::1          | icmp |   135 |         0 |
| fe80::b44b:2aef:e6f6:c6ce | ff02::16         | icmp |     0 |       440 |
| fe80::b44b:2aef:e6f6:c6ce | ff02::1:2        | udp  |   547 |       783 |
| fe80::b44b:2aef:e6f6:c6ce | ff02::1:3        | udp  |  5355 |        50 |
| fe80::b44b:2aef:e6f6:c6ce | ff02::1:fff6:c6ce | icmp |   136 |         0 |
| fe80::b44b:2aef:e6f6:c6ce | ff02::2          | icmp |   134 |        16 |
+--------------------------+------------------+------+-------+-----------+
6 rows in set, 10 warnings (0.00 sec)
```

2.2.4. Next Steps for Consideration

The use of this homegrown code and query examples would obviously not suffice in a production environment. However the concept can be taken and applied to more mature asset/configuration management solutions in use at an organization. There is no lack of commercial and open source options, so there is an opportunity to advance the concepts presented in this paper by attempting to integrate network layer metadata into an existing configuration management solution. For example, chef is open source configuration management software that provides an application programming interface (API).

Similarly, Bro has worked well in this example, and indications are that it may support direct logging to a MySQL database in the future. But there may be a more suitable mechanism for capturing network conversation information and storing it for later presentation in the asset and configuration centric views. Additional research is required in this area.

Finally, additional consideration needs to be given to specifics of what network metadata would be presented and how. For example, thresholds around session duration, expected byte/packet counts, session state (fully established vs. attempted) may be difficult to pinpoint in terms of what is considered normal. Initially, it may be more

valuable to simply present such information in the context of a given asset so as to help the system owner/admin decide which conversations are authorized/approved.

3. Conclusion

Increased use of encryption to protect confidentiality of network transmissions, both internally and across public networks, is an impetus for evolving how network layer security analysis is performed. One response to this is to consider integration of unencrypted network layer conversation metadata, and characteristics of the nodes involved, with asset and configuration management databases/baselines. Doing so provides an asset-centric control and enhances the opportunity for review and authorization/approval of network communications. In organizations that leverage good configuration management as a security control, it serves as an additional layer of detection of unauthorized network activity. The concept and working examples presented can be advanced by integrating with an existing asset/configuration management to further demonstrate the value of view network connectivity information as a discrete configuration item.

William Yeatman, wmyeatman@gmail.com

4. References

Bejtlich, R. (2005). The Tao of Network Security Monitoring: Beyond Intrusion Detection. Addison-Wesley.

Filkins, B. (2014). *Health Care Cyberthreat Report: Widespread Compromises Detected, Compliance Nightmare on Horizon*. Retrieved from http://www.sans.org/reading-room/whitepapers/firewalls/health-care-cyberthreat-report-widespread-compromises-detected-compliance-nightmare-horizon-34735

Finley, K. (2014, May 16). Encrypted Web Traffic More Than Doubles After NSA Revelations | WIRED. Retrieved from http://www.wired.com/2014/05/sandvine-report/

Grosfelt, J. (2014, January 15). Command and Control Encryption – Part [RSA blog post]. Retrieved from https://blogs.rsa.com/command-control-encryption-part-1/

Karwaski, M. (2009). *Efficiently Deducing IDS False Positives Using System Profiling*. Retrieved from http://www.sans.org/reading-room/whitepapers/detection/efficiently-deducing-ids-false-positives-system-profiling-33223

Ponemon Institute. (2014). *Global Encryption Trends Study*. Retrieved from https://www.thales-esecurity.com/knowledge-base/analyst-reports/global-encryption-trends-study

Sandvine. (2014). *Global Internet Phenomena Report: 1H 2014*. Retrieved from Sandvine website: https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/1h-2014-global-internet-phenomena-report.pdf

SANS. (2014). SANS Institute - Critical Security Control: 1. Retrieved from http://www.sans.org/critical-security-controls/control/1

SANS. (2014). SANS Institute - Critical Security Control: 2. Retrieved from http://www.sans.org/critical-security-controls/control/2

Scott, C. (2014). *Designing and Implementing a Honeypot for a SCADA Network*. Retrieved from http://www.sans.org/reading-room/whitepapers/detection/designing-implementing-honeypot-scada-network-35252

William Yeatman, wmyeatman@gmail.com

Appendix

broconns.py listing

```
from os import listdir
import re
import mysql.connector
import socket
import ipaddr
#import struct
import gzip
import geoip2.webservice
from IPy import IP
import logging

logging.captureWarnings(True)

brologpath = "/home/code/bro/logs/"
p = re.compile('conn\..*\.gz$')  # match conn. at the beginning of filename
l = re.compile('^[^#]')   # want to ignore # comment lines in bro files

mysql_config = {
    'host': '127.0.0.1',
    'user': 'sans',
    'password': 'sans',
    'database': '***********',
    #  'raise_on_warnings': True
}

cnx = mysql.connector.connect(**mysql_config)
cursor = cnx.cursor()

for dir in listdir(brologpath):
    for filename in listdir(brologpath + dir):
        if p.match(filename):
            fullpath = brologpath + dir + "/" + filename
            fpqry = ("SELECT filename FROM processed_files WHERE filename=\'" + fullpath
+ "\'")
            print(fpqry)
            cursor.execute(fpqry)
            row = cursor.fetchone()
            if row is not None:
                print "That file was already processed, skipping"
            else:
                print "That file does not appear to have been processed yet"

                f = gzip.open(fullpath)
                for line in f:
                    if l.match(line):
                        fields = line.rsplit()

                        ts = fields[0]
                        sip_str = fields[2]
                        dip_str = fields[4]
                        dport = fields[5]
                        prot = fields[6]
                        svc = fields[7]
                        dur = fields[8]
                        orig_bytes = fields[9]
                        resp_bytes = fields[10]
                        orig_pkts = fields[16]
                        resp_pkts = fields[18]

                        sip = ipaddr.IPAddress(sip_str)
```

William Yeatman, wmyeatman@gmail.com

```
dip = ipaddr.IPAddress(dip_str)
if sip.version == 4 & dip.version == 4:
    conn_data = {
        'sip4': sip_str,
        'dip4': dip_str,
        'prot': prot,
        'dport': dport,
        'svc': svc
    }
    add_conn = ("INSERT INTO conn4 "
                "(sip4,dip4,prot,dport,svc,ts,"
                "duration, orig_bytes, resp_bytes, "
     "orig_pkt, resp_pkt) "
                "VALUES(%(sip4)s,%(dip4)s,%(prot)s,"
                "%(dport)s,%(svc)s,%(ts)s,%(dur)s,"
                "%(orig_bytes)s, %(resp_bytes)s, "
                "%(orig_pkts)s, %(resp_pkts)s)")

elif sip.version == 6 & dip.version == 6:
    conn_data = {
        'sip6': sip_str,
        'dip6': dip_str,
        'prot': prot,
        'dport': dport,
        'svc': svc
    }
    add_conn = ("INSERT INTO conn6 "
                "(sip6, dip6, prot, dport, svc, ts, "
                "duration, orig_bytes, resp_bytes, "
     "orig_pkt, resp_pkt) "
                "VALUES (%(sip6)s, %(dip6)s, %(prot)s, "
                "%(dport)s, %(svc)s, %(ts)s, %(dur)s, "
                "%(orig_bytes)s, %(resp_bytes)s, "
                "%(orig_pkts)s, %(resp_pkts)s)")

else:
    print("Mismatched ip versions detected")

stats_data = {
    'ts': ts,
    'dur': dur,
    'orig_bytes': orig_bytes,
    'resp_bytes': resp_bytes,
    'orig_pkts': orig_pkts,
    'resp_pkts': resp_pkts
}
conn_data.update(stats_data)

cnx = mysql.connector.connect(**mysql_config)
cursor = cnx.cursor()

try:
    cursor.execute(add_conn, conn_data)
except mysql.connector.Error as err:
    print("error: ")
    print(err)
else:
    cnx.commit()

# get intel on the sip, and dip
# if it is rfc1918, then just do hostname

if dip.version == 4:
    qry = ("SELECT ip4 FROM ip4_intel "
            "WHERE ip4 = \'" + dip_str + "\'")
    print qry
    cursor.execute(qry)
    row = cursor.fetchone()
    if row is not None:
        print "IP already in intel table, no geo query needed"
```

```
                            else:
                                print "IP not in intel table, better do the lookup and
insert"
                                ip = IP(dip_str)
                                ip_type = ip.iptype()
                                if ip_type == 'PRIVATE':
                                    #  do a hostname lookup
                                    print "DEBUG: in PRIVATE section"
                                    #name, alias, addresslist =
socket.gethostbyaddr(dip_str)
                                    print "*** " + dip_str + "***"

                                    add_intel = ("INSERT INTO ip4_intel "
                                        "(ip4) VALUES (%(dip_str)s)")
                                    intel_data = {
                                        'dip_str': str(dip_str)
                                    }
                                    try:
                                        cursor.execute(add_intel, intel_data)
                                    except mysql.connector.Error as err:
                                        print("error: ")
                                        print(err)
                                    else:
                                        cnx.commit()


                                elif ip_type == 'PUBLIC':
                                    #  get the geo info
                                    print "DEBUG: in PUBLIC section"
                                    geoclient = geoip2.webservice.Client(#####, '*****',
'geoip.maxmind.com')
                                    response = geoclient.city(dip_str)
                                    asn = response.traits.autonomous_system_number
                                    asn_org =
response.traits.autonomous_system_organization
                                    org = response.traits.organization
                                    isp = response.traits.isp
                                    dom = response.traits.domain
                                    country = response.country.name
                                    city = response.city.name
                                    queries_left = response.maxmind.queries_remaining

                                    print "remaining queries: " + str(queries_left)
                                    add_intel = ("INSERT INTO ip4_intel "
                                        "(ip4, geoip_asn_num, geoip_asn_org, geoip_org, "
                                        " geoip_isp, geoip_dom, geoip_country,
geoip_city) "

                                        " VALUES (%(dip_str)s, %(asn)s, %(asn_org)s, "
                                        " %(org)s, %(isp)s, %(domain)s, %(country)s, "
                                        " %(city)s)")
                                    intel_data = {
                                        'dip_str': str(dip_str),
                                        'asn': asn,
                                        'asn_org': str(asn_org),
                                        'org': str(org),
                                        'isp': str(isp),
                                        'domain': str(dom),
                                        'country': str(country),
                                        'city': city
                                    }
                                    try:
                                        cursor.execute(add_intel, intel_data)
                                    except mysql.connector.Error as err:
                                        print("error: ")
                                        print(err)
                                    else:
                                        cnx.commit()
                                else:
                                    print str(ip.iptype) + "found"
                        cursor.close()
```

William Yeatman, wmyeatman@gmail.com

```
        cnx.close()

#insert filename
cnx = mysql.connector.connect(**mysql_config)
cursor = cnx.cursor()
add_file_processed = ("INSERT INTO processed_files "
            "(filename) VALUES(\'" + fullpath + "\')" )
print "qry: " + add_file_processed
fname_data = {
    'fullpath': fullpath
    }
try:
    cursor.execute(add_file_processed)
except mysql.connector.Error as err:
    print("error: ")
    print(err)
else:
            cnx.commit()
```


William Yeatman, wmyeatman@gmail.com