NetFlow collection and analysis using nfcapd, Python, and Splunk

GIAC GCIA Gold Certification

Author: David Mashburn dmashburn@gmail.com
Advisor: Rob VandenBrink

Accepted: 15 January 2015

Abstract

NetFlow data is often collected for network monitoring and management, but it has many applications for the security analyst. NetFlow data can be used to identify variations from established traffic baselines, traffic originating from critical systems, and communications with known bad external hosts. Many edge devices support the generation of NetFlow data, but the collection and analysis often requires commercial tools. Options based on open source and free tools will allow an analyst to deploy a solution when commercial solutions are not available due to budgetary or other organizational constraints. Solutions based on this proposed architecture will provide the required NetFlow data collection resources that can feed a variety of systems that could be leveraged for analysis. The reference architecture described utilizes Ubuntu server, the nfdump suite of tools, custom Python scripts, and the free version of a commercial tool for analysis.

1. Introduction

NetFlow is a traffic summary technology developed by Cisco systems. While intended as a management and auditing tool for networking professionals, NetFlow data can be a valuable resource for security analysts. Typically generated by edge devices such as a router or a firewall, NetFlow data summarizes traffic and "provides valuable information about network users and applications, peak usage times, and traffic routing" ("Cisco IOS NetFlow – Cisco", n.d.). NetFlow does not provide the full content of a packet, as it does not store the packet payload. NetFlow provides visibility into communication details such as the source IP address, destination IP address, protocol, source and destination ports for TCP ad UDP traffic, and bytes transferred. This limited footprint for NetFlow in terms of storage requirements and processing resources allows for the use of this technology in situations where a packet sniffer would not be suitable (Scheck, 2009).

NetFlow can be used for baseline monitoring of network activity at any designated point on the network, but typically NetFlow data is generated on edge devices to provide information about network traffic to external destinations and as well as the sources of inbound traffic. NetFlow also provides visibility into the volume of network traffic flows, so the context provided by NetFlow data is very useful to a security analyst.

NetFlow data can provide "network situational awareness" for security analysts (Chickowski, 2013). This can range from observing indications of system compromise to unusual traffic volumes to identifying possible shadow IT that may exist within an organization (Chickowski, 2013). NetFlow is often used to follow up on alarms or alerts from other systems such as an Intrusion Detection System (IDS) (Patterson, 2012). NetFlow provides significant value to an analyst in the area of threat detection based on Internet Protocol (IP) address reputation (Patterson, 2012). NetFlow is a valuable addition to the tools available to a security analyst, as it can be used to provide temporal and traffic context around a security event.

NetFlow data should be considered as a tool to enrich the security analysis and incident response process rather than a primary system to detect threats. Patterson (2012) states that systems relying on NetFlow analysis are "only suitable as a second or perhaps third layer of threat detection." Considering that Patterson is CEO of a vendor that offers commercial NetFlow tools, that advice should heeded as warning to properly use the NetFlow collection and analysis in a complementary role in security operations.

When properly utilized in a supporting role, NetFlow data can be a powerful tool for the security analyst.

1.1. NetFlow data specifications

Edge devices must be configured to generate NetFlow data. NetFlow data can be exported from the edge device in several different versions. The NetFlow record specification has evolved over time, with several versions currently in wide use. Currently version 5 and version 9 are the most common. Version 9 is the basis for the IETF standard IPFIX that is described in RFC 7011 (Claise, Trammell, & Aitken, 2013). NetFlow version 5 uses a fixed format for records, while NetFlow version 9 supports templates that provide more flexibility in terms of defining the fields to be included in the exported data (Patterson, 2013). Another significant difference between NetFlow version 5 and NetFlow version 9 is the support for IPv6. Only NetFlow v9 supports IPv6, so as organizations embrace IPv6 this version will become more common. However, NetFlow version 5 is still often used due to the legacy installation footprint of tools and software.

The minimum information needed to configure the edge device is the destination IP address, the destination UDP port, and the NetFlow version that will be utilized. There typically is some performance overhead on the device associated with enabling NetFlow output (Scheck, 2009), so the edge device must have sufficient processing and bandwidth capacity to handle the increased load.

1.2. Requirements

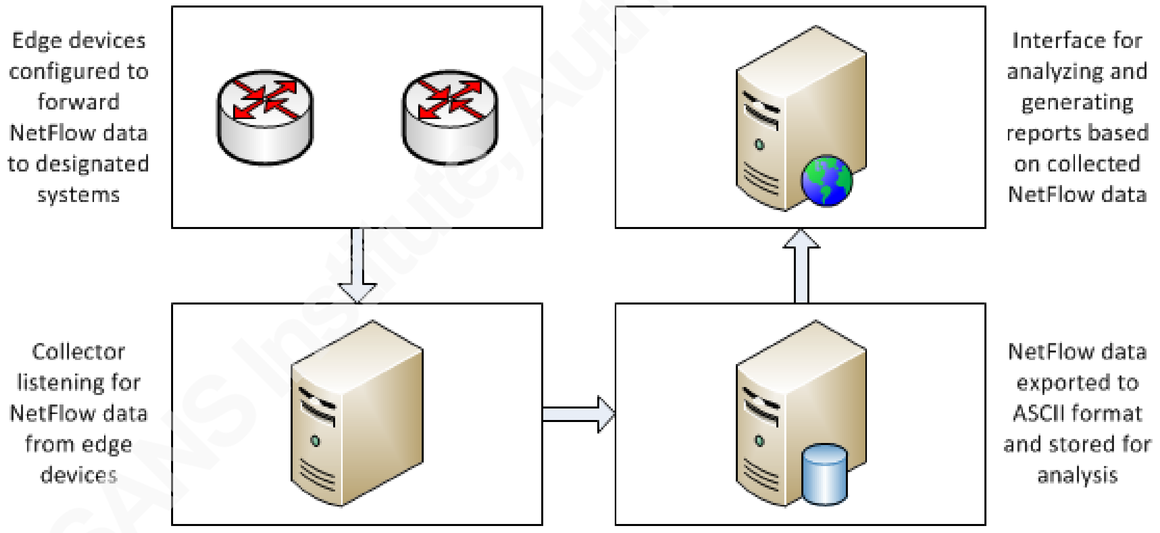
The proposed solution must collect NetFlow data generated by an edge device in either the version 5 or the version 9 specification. In addition, the binary format NetFlow data must be converted to a format suitable for import into the desired analysis tool. The native logs as well as the exported logs should be rotated and periodically purged.

Finally, the analysis tool must consume the data and allow for development of custom queries, reports, and visual representations of the flow data to facilitate analysis.

2. Solution overview

The reference architecture described is composed of loosely coupled components that allow for the substitution of different solutions for each tier. The three tiers are data collection, data management, and data analysis. For ease of deployment, in this reference architecture, all of the tiers will reside on the same system. However, in other deployment scenarios, each tier does not necessarily need to reside on the same host. The reference architecture is based on deployment on a 64-bit Ubuntu Server version 14.04.1 LTS.

2.1. Logical diagram


Edge devices configured to forward NetFlow data to designated systems
Interface for analyzing and generating reports based on collected NetFlow data
Collector listening for NetFlow data from edge devices
NetFlow data exported to ASCII format and stored for analysis

2.2. Data collection

NetFlow data is generated in a binary format. The collector must be able to receive the data and place it in local storage for processing. The open source tool nfcapd from the nfdump suite of utilities will handle the data collection in this tier.

2.3. Data management

The data collection tier will create files that contain the binary NetFlow data. The data management tier will handle the export of the data to a suitable format for the analysis engine as well as the file rotation, archiving, and cleanup. The nfdump utility will be used to export the NetFlow data into ASCII text, which is the preferred format for the analysis engine utilized in this design.

2.4. Data analysis

The data analysis tier will consume the files that are processed by the data management tier. The system must then be able to support reporting, including visualizations, so that useful and actionable intelligence can be created from the NetFlow data. Splunk Enterprise is the tool of choice for this tier. Splunk is well-suited for the analysis of time-series data and provides many visualization options, making it an ideal choice for this role. Splunk has a licensing model based on the volume data indexed each day, and many organizations may already have sufficient capacity available in their licensed volume to accommodate NetFlow data. For those without a current license, a free license is available for daily index volume of 500 MB ("Free vs. Enterprise | Splunk License Comparison Table | Splunk," n.d.). This may be suitable for smaller organizations and should be sufficient for many lab or test environments.

3. NetFlow data collection

3.1. Edge device configuration

The configuration of edge devices such as routers or firewalls must be updated to send NetFlow data to the nfcapd listener. This is listed as a first step as it may take more time than the configuration of the other items, as the security analyst may lack the access needed to perform the configuration changes. The security analyst will need to provide the IP address, destination UDP port, and the desired NetFlow version to those configuring the devices to export NetFlow data.

A key consideration for the devices selection for export is Network Address Translation (NAT). Many firewalls use a single external IP address to mask internal addresses, so NetFlow data collected from a firewall or router outside the firewall will not provide the specific internal IP address that is accessing an external resource. The ideal NetFlow export point is a router that forwards traffic to the firewall or a firewall that can export NetFlow data prior to the NAT processing. NetFlow analysis of data is still possible if data is only available after NAT, but the trace back to the internal source cannot be done solely with the NetFlow data.

Consult the documentation of your specific devices for the commands required to enable NetFlow export.

3.2. Nfcapd installation

The current version of the nfdump suite is 1.6.13 as of 1 December 2014. The source archive should be downloaded from the nfdump project page on SourceForge. After downloading, extract the archive to your designated host. The host requires make, gcc, and flex in order to compile the nfdump utilities, so you may need to install additional packages to make sure that all of the prerequisites are met prior to compilation.

The commands below show the required steps to compile and install nfcapd. The utilities are installed in /usr/local/bin. The commands shown below were run in the home directory of the current user, but may be run in any suitable directory where the current user has write privileges.

```
user@host:~$ tar zxvf nfdump-1.6.13.tar.gz
user@host:~$ cd nfdump-1.6.13
user@host:~/nfdump-1.6.13$ ./configure
user@host:~/nfdump-1.6.13$ make
user@host:~/nfdump-1.6.13$ sudo make install
```

The use of sudo is required for the final step in order to place the compiled binaries in /usr/local/bin.

4. NetFlow data management

The deployment of the nfdump tools is the first step in creating a collection point for NetFlow data. The listener must be configured, started, and monitored on the designated server. The binary NetFlow logs must be converted to a format suitable for ingestion by the analysis tier. The lifecycle of the data files must also be addressed so that the NetFlow data collection and export process cleans up after itself and does not overwhelm the local filesystem. Each of these tasks is accomplished through the use of the native cron scheduler and a script.

4.1. Configuration with Python scripts

A set of scripts to configure the nfcapd listener, the log rotation jobs, and the log archive management were developed in the Python language. The host system will need Python version 2 to run the scripts. Optionally, Git can be used to retrieve the scripts from the code repository. These Python scripts are available on GitHub and are licensed under the GNU GPL, so they can be used as-is or modified as needed for specific needs. The scripts should be run using sudo as they write to file system locations that are not typically available to standard users. The scripts will create files and folders in the /opt directory.

```
user@host:~$ cd /opt/
user@host:/opt$ sudo git clone https://github.com/kravp00L/netflow.git
```

Once the repository has been cloned, the scripts will be in the netflow folder that is created by the cloning process. The configuration script should be executed in order to configure the system. This script will take interactive input to configure the NetFlow listener(s), log rotation and retention settings, and verify the locations of the required components. The script creates a directory for data and a directory for the configuration files.

```
user@host:/opt$ cd netflow
user@host:/opt/netflow$ sudo python ./configure.py
```

The crontab must be updated so that the required scripts are run. By default, the listener script runs every five (5) minutes to make sure that nfcapd is active. If it is not active it starts one or more listeners based on the configuration that was created with the configure.py script. The log dump script runs every minute to export any unprocessed binary log files to ASCII. The binary logs are then placed in the archive directory after processing by the script. The log maintenance script runs once per day to remove any logs that are beyond the retention interval. These scripts will be run when you add the contents of conf/cron.entries into the crontab for the desired user. While this example runs the scripts and listener as root, you can use any account with sufficient permissions to create and to write to the /opt/netflow directory. For production deployment it would be preferable to run under a non-root account.

The table below describes the function of each of the scripts and the default schedule associated with the script that is found in the /opt/netflow/conf/cron.entries file. The default schedule can be adjusted to suit your specific needs when you add the entries to crontab.

Script name	**Script function**	**Default schedule**
configure.py	Create configuration files that will be used by the scheduled jobs.	Run once
log_cleanup.py	Deletes files that are past the retention interval as specified during setup.	Executes once daily
log_dump.py	Exports the binary flow logs to ASCII and performs file archiving.	Executes every minute
netflow.py	Starts and monitors the nfcapd instances as specified during setup.	Executes every 5 minutes
read_params.py	Debugging script to verify parameters entered during setup.	None

4.2. Cron configuration

To minimize the possibility of disrupting existing cron jobs as well as to allow for setup of this environment using a non-root user, the required cron jobs are not automatically entered in the crontab. A file is written in the conf directory named cron.entries. A simple copy and paste of the lines from this file into the crontab will schedule the scripts described in the previous section.

```
user@host:/opt/netflow/conf$ head cron.entries
```

```
# Add the entries below to crontab for root

*/5 * * * * python /opt/netflow/netflow.py

*/1 * * * * python /opt/netflow/log_dump.py

30 22 * * * python /opt/netflow/log_cleanup.py
```

5. NetFlow analysis

The final and most important part of the handling of NetFlow data is the ability to perform analysis on the flow data. The reference architecture presented provides loose coupling between the various tiers, so that the collection tool could be replaced with a different software tool. The same is true for the data feed to the analysis system. The choice of analysis tool presented here is one example of what could be used, but any system capable of consuming the data generated from the NetFlow collection tier will be suitable. ELSA may be a suitable choice for those who have an existing deployment of the Security Onion, while the ELK (elasticsearch-logstash-kibana) stack may be an appropriate choice for other organizations. In this case, Splunk Enterprise is the chosen analytics platform. Splunk is well-suited for time-series data, provides a rich set of analytic functions, and has excellent visualization capabilities for creating rich output such as dashboards or reports.

5.1. Splunk Enterprise overview

Splunk Enterprise logically separates the data collection and indexing process from the searching and reporting features. There are multiple roles defined in a Splunk deployment. A forwarder sends data to a remote indexer. An indexer is a system which stores and processes the data feeds from the local filesystem or from one or more remote systems. The search head is the part of Splunk that handles the reporting and analysis functions by directing queries to the appropriate indexer ("Components of a Splunk Enterprise deployment," n.d.). While logically separate, these roles can readily exist on the same host. This single host deployment where the forwarder, indexer, and search head are on same host is the one that will be utilized in this reference architecture.

5.2. Splunk Enterprise installation

Splunk Enterprise is licensed based on the amount of data that is indexed daily across all indexers in a given deployment. The free license of Splunk is utilized in this reference architecture, so 500 MB of NetFlow data can be indexed daily.

The command below installs Splunk Enterprise on the local system. Splunk can only be installed to /opt/splunk when using the Debian package ("Install on Linux," n.d.). The splunk binary controls the application and is started as shown below:

```
user@host:~$ sudo dpkg -i splunk-6.2.1-245427-linux-2.6-amd64.deb
user@host:~$ sudo /opt/splunk/bin/splunk start --accept-license
user@host:~$ sudo /opt/splunk/bin/splunk enable boot-start
```

The --accept-license option is only necessary the first time the application is started. In addition to starting the Splunk instance, the last command configures the system to automatically start Splunk after system boot. Additional configuration of Splunk is required to get the system to consume the NetFlow data that has been captured by nfcapd and converted to ASCII text by nfdump.

Splunk, once started, runs a web server on port 8000. Open up the Splunk web interface in a web browser. The default credentials for the web interface are admin / changeme for Splunk 6.2.1. The default password is required to be changed at the first login to the web interface. The admin password is required for command line operations, so this step must be completed prior to further configuration.

If you prefer to change the admin password at the command line, use the following command, where <newpassword> is replaced with the desired admin password:

```
user@host:/opt/splunk/bin$ sudo ./splunk edit user admin -password
<newpassword> -role admin -auth admin:changeme

User admin edited.
```

5.3. Splunk index creation

The Splunk indexer contains an index named main. All data is stored in this main index by default unless otherwise specified. In order to keep the NetFlow data separate from other unrelated data, an index specifically for NetFlow data must be created on the indexer. Execute the command below must to create the new index.

```
user@host:/opt/splunk/bin$ sudo ./splunk add index netflow -auth
admin:<password>
Index "netflow" added
```

After the netflow index has been created on the Splunk indexer, additional configuration to consume and store data in the netflow index must be completed for the local Splunk instance.

5.4. Splunk configuration

Splunk can be configured through the web interface, the command-line, or through the use of text-based configuration files. There are several essential settings that must be configured in Splunk to allow the NetFlow data to be consumed.

Splunk uses a set of text configuration files to control the behavior of the application. By convention, these files are located in $SPLUNK_HOME/etc/system in one of two folders. The default folder contains the base configuration files and those files should not be altered. The local folder is initially empty, but that is the location for the local configuration settings for the system. The configuration files in the local folder will override the settings of those in default folder.

There are two files that must be created (or edited if already present) in $SPLUNK_HOME/etc/system/local in order for data to be indexed by Splunk: inputs.conf and outputs.conf. The inputs.conf file defines the data sources that will be monitored by Splunk, while outputs.conf provides details about the receiving Splunk instance.

The settings below show the entries that need to be made in inputs.conf so that Splunk will index the NetFlow data collected by the nfcapd listener. If deploying to an existing Splunk system, the stanzas show below can be appended to the local inputs.conf file. Otherwise, the inputs.conf file generated by the configure.py script can be copied from /opt/netflow/conf to $SPLUNK_HOME/etc/system/local without requiring any changes.

To configure the local Splunk instance to consume the exported NetFlow data, use the following command:

```
user@host:/opt/splunk/bin$ sudo ./splunk add monitor
/opt/netflow/data/nfdump-ascii/ -index netflow –sourcetype netflow

Added monitor of '/opt/netflow/data/nfdump-ascii'.
```

For a distributed Splunk instance, the following entries can be copied to the $SPLUNK_HOME/etc/system/local/inputs.conf:

```
# input stanzas for NetFlow data
[monitor:///opt/netflow/data/nfdump-ascii]
index = netflow
sourcetype = netflow
disabled = false
```

In addition to defining the data to be indexed by Splunk, the destination Splunk indexer must be also specified. This is done in the outputs.conf file. In this reference architecture, changes to outputs.conf are not required since all of the Splunk services reside on the same host. However, in a distributed deployment of Splunk Enterprise, the file would need to specify the remote Splunk indexer in use.

```
# output stanza for remote Splunk indexer
[tcpout]
defaultGroup = netflow_splunk
useACK = true

[tcpout:netflow_splunk]
server = <Remote Splunk indexer IP address>:9997
```

David Mashburn, dmashburn@gmail.com

Replace <Remote Splunk indexer IP address > with the IP address or hostname of the destination Splunk indexer. If you use a hostname make sure that the name resolves correctly or the data will not make it to the indexer.

5.5. Splunk source type definition

After Splunk has been configured to accept the NetFlow data in the netflow index, the data format must be defined so that Splunk can extract the values into discrete fields. Since nfdump generates output in CSV format, it is very simple to parse the data. The only information needed is the field order of the data file. Some of the fields in the CSV output may not be of interest or may not be populated in the export based on your particular environment, so there may be a number of fields that are zero valued.

Two files are needed to handle the source type definition: props.conf and transforms.conf. Each of the files will reside in the $SPLUNK_HOME/etc/system/local directory with the other local configuration files.

The configuration script creates a props.conf file in /opt/netflow/conf that can be used, or on the local Splunk indexer, create a props.conf file in the $SPLUNK_HOME/etc/system/local directory as follows:

```
# definition for netflow sourcetype
[netflow]
CHECK_FOR_HEADER = false
SHOULD_LINEMERGE = false
REPORT-netflow_field_extract = netflow_csv
FIELDALIAS-src = src_ip AS src
FIELDALIAS-dst = dst_ip AS dst
FIELDALIAS-input_bytes = input_bytes AS bytes
```

The configuration script creates a props.conf file in /opt/netflow/conf that can be used, or on the local Splunk indexer, create a transforms.conf file in the $SPLUNK_HOME/etc/system/local directory as follows:

```
# stanza for netflow extractions

[netflow_csv]

DELIMS = ","

FIELDS = "flow_start_time", "flow_end_time", "flow_duration",
"src_ip", "dst_ip", "src_port", "dst_port", "protocol", "tcp_flag",
"fwd_status", "src_tos", "input_pkts", "input_bytes", "output_pkts",
"output_bytes", "in_if", "out_if", "src_bgp_as", "dst_bgp_as",
"src_mask", "dst_mask", "dst_tos", "flow_dir", "next_hop_rtr",
"bgp_next_hop_rtr", "src_vlan", "dest_vlan", "in_src_mac",
"out_dst_mac", "in_dst_mac", "out_src_mac", "mpls1", "mpls2",
"mpls3", "mpls4", "mpls5", "mpls6", "mpls7", "mpls8", "mpls9",
"mpls10", "client_latency", "server_latency", "app_latency",
"rtr_ip", "engine", "exp_sys_id", "flow_received"
```

The combination of props.conf and transforms.conf will extract the discrete fields and give them the correct field names in the Splunk web interface at search time. The fields will appear under the list of Interesting Fields in the web interface, and can be directly referenced in a search.

Please note that the configuration steps described in sections 5.3, 5.4, and 5.5 can also be completed through the Splunk web interface. These configuration settings are reloaded automatically and do not require a restart of the Splunk services.

5.6. Splunk queries

Once Splunk has been configured to index the NetFlow data into a dedicated index, the data can be leveraged by the security analyst in the incident handling process. Queries for specific traffic based on source IP address, destination IP address, or other characteristics of interest can now be generated. The netflow project on GitHub contains sample Splunk reports and dashboards that can be used by security analysts. A few examples of possible report are detailed in the following section and presented as a use case for the data.

Searches are saved as Reports in Splunk parlance. The term query is used at times to describe searches in Splunk. However, using the term query some to attempt to replicate relational SQL queries in Splunk, which really don't directly map to the search language. A better approach is to think in terms of grep, where large data sets are filtered quickly to leave only the results of interest. Splunk searches filter the results and pipe the result set to the next command for further evaluation or refinement.

6. NetFlow Data Use Cases

With the NetFlow data now being collected by nfcapd, exported by nfdump, and indexed by Splunk, a security analyst can generate reports to assist in investigations and incident response. Some common use cases for NetFlow data are covered in the following sections, along with the corresponding Splunk search string.

6.1. Top talkers

The searches below generates a list of the 25 most active hosts by bytes transferred for the previous day and for the past 24 hours to the nearest full hour. In many environments, the traffic should be primarily flowing inbound to reflect common web usage patterns. An internal host that is the source of large amounts of data transfer should be investigated.

```
# yesterday

earliest=-1d@d index=netflow | stats sum(bytes) as Bytes by src_ip,
dst_ip | eval MB=Bytes/(1024*1024)| rename src_ip as Source, dst_ip
as Destination | table Source, Destination, MB | sort 25 -MB


# last 24 hours

earliest=-1d@h index=netflow | stats sum(bytes) as Bytes by src_ip,
dst_ip | eval MB=Bytes/(1024*1024)| rename src_ip as Source, dst_ip
as Destination | table Source, Destination, MB | sort 25 -MB
```

6.2. Top protocols

The searches below generate a list of the active protocols by bytes transferred for the previous day and for the past 24 hours to the nearest full hour. This is another set of baseline queries that can be used to identify unusual protocol occurrences or volumes.

```
# yesterday

earliest=-1d@d index=netflow | stats sum(bytes) as Bytes by protocol
| eval MB=Bytes/(1024*1024)| rename protocol as Protocol | table
Protocol, MB | sort -MB


# last 24 hours

earliest=-1d@h index=netflow | stats sum(bytes) as Bytes by protocol
| eval MB=Bytes/(1024*1024)| rename protocol as Protocol | table
Protocol, MB | sort -MB
```

6.3. Top TCP services

The searches below generate a list of the TCP activity. This is where the analyst has to apply knowledge of the environment, as services may run on non-standard ports. The search captures presumed responses or downloads and presumed requests or uploads based on the source and destination ports. Traffic to ports below 1024 is presumed to be services requests and traffic to ports above 1024 is presumed to be responses. The analysis will need to validate that guidance in the context of the actual traffic.

```
#presumed responses or downloads for past hour

earliest=-1h@h index=netflow TCP | where src_port < 1024 AND
dst_port > 1023 | stats sum(bytes) as Bytes by src_port, dst_port |
eval MB=Bytes/(1024*1024) | rename src_port as Source, dst_port as
Destination | table Source, Destination, MB | sort -MB


#presumed requests or uploads for past hour

earliest=-1h@h index=netflow TCP | where dst_port < 1024 AND
src_port > 1023 | stats sum(bytes) as Bytes by src_port, dst_port |
eval MB=Bytes/(1024*1024) | rename src_port as Source, dst_port as
Destination | table Source, Destination, MB | sort -MB
```

6.4. Insecure protocol usage

The search below looks for well-known insecure services on their standard ports. Items of interest include ftp [TCP ports 20 and 21], telnet [TCP port 23], tftp [UDP port 69], rexec [TCP port 512], rlogin [TCP port 513], and rsh [TCP port 514]. This search could also be modified to look for any application activity of interest based on the source or destination ports.

```
# activity last 4 hours

earliest=-4h@h index=netflow | where match(src_port,
"^(20|21|23|69|512|513|514)$") OR match(dst_port,
"^(20|21|23|69|512|513|514)$") | stats sum(bytes) as Bytes by
src_ip, dst_ip, src_port, dst_port | eval MB=Bytes/(1024*1024)|
rename src_ip as Source, dst_ip as Destination, src_port as SPort,
dst_port as DPort | table Source, Destination, SPort, DPort, MB |
sort 25 -MB
```

This search can be easily modified to baseline traffic to or from critical systems. Add in a filter to a specific source or destination IP address before the where clause and update the regular expression for the port list.

6.5. IP geolocation

The search below gets a count of the number of occurrences of each destination IP address and then performs a lookup to determine the geographic location of the destination IP address. Internal [RFC 1918] IP addresses will return no information. This can help quickly identify hosts calling out to locations that might be considered suspicious. This search can be saved as a report and embedded into a Dashboard so that the destination IP addresses are overlaid on a map.

```
# geolocation lookup for destination IP addresses

index=netflow earliest=-1h@h | stats count(dst_ip) as Count by
dst_ip | iplocation dst_ip | rename dst_ip as Destination | table
Destination, Count, City, Country | sort -Count
```

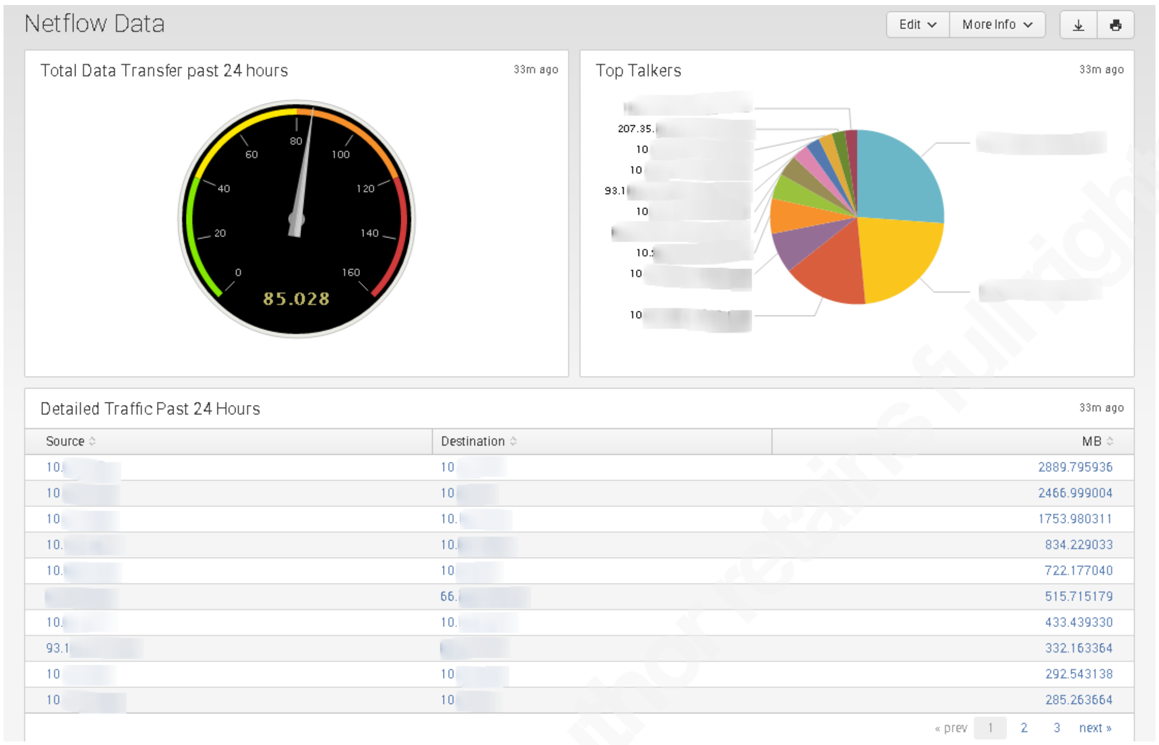
6.6. Dashboard example

Splunk provides many options for data visualization. In the sample dashboard shown below, there are three panels that demonstrate a subset of the available visualizations. While an analyst needs access to raw data, dashboards are useful to show trends or summary of activity. The dashboard uses a pie chart, a table, and a radial gauge to present data of interest. Each of these panels in the dashboard is backed by a search that returns the required data, and the panel configuration specifies the desired visualization for the data.

Dashboard configuration is stored in an XML format. The search string can be embedded in the XML or it can refer to a saved report. For clarity, the example dashboard embeds the search string in the XML. The radial gauge shows the total GB of data transferred in the past 24 hours. The pie chart lists the top twelve most active talkers in the past 24 hours by source and destination IP address. The detailed traffic flow adds in the destination port as an additional field to provide additional insight into the traffic flows but does not display the port since many of the destination ports are in the ephemeral port ranges and are reflective of return traffic from web requests.

```
# XML for dashboard
       <dashboard>
         <label>Netflow Data</label>
         <row>
           <chart>
             <title>Total Data Transfer past 24 hours</title>
             <searchString>index=netflow | stats sum(bytes) as Bytes |
eval GB=Bytes/(1024*1024*1024) | fields GB</searchString>
             <earliestTime>-24h@h</earliestTime>
             <latestTime>now</latestTime>
             <option
name="charting.axisTitleX.visibility">visible</option>
             <option
name="charting.axisTitleY.visibility">visible</option>
             <option name="charting.axisX.scale">linear</option>
             <option name="charting.axisY.scale">linear</option>
```

```
            <option name="charting.chart">radialGauge</option>
            <option name="charting.chart.nullValueMode">gaps</option>
            <option
name="charting.chart.rangeValues">["0","40","80","120","160"]</option>
            <option
name="charting.chart.sliceCollapsingThreshold">0.01</option>
            <option name="charting.chart.stackMode">default</option>
            <option name="charting.chart.style">shiny</option>
            <option name="charting.drilldown">all</option>
            <option
name="charting.gaugeColors">[0x84E900,0xFFE800,0xf7912c,0xd13b3b]</option>
            <option name="charting.layout.splitSeries">0</option>
            <option
name="charting.legend.labelStyle.overflowMode">ellipsisMiddle</option>
            <option name="charting.legend.placement">right</option>
          </chart>
          <chart>
            <title>Top Talkers</title>
            <searchString>index=netflow | stats sum(bytes) as Bytes by
src_ip, dst_ip, dst_port | eval MB=Bytes/(1024*1024)| eval
Traffic=src_ip+" &gt; "+dst_ip | table Traffic, MB | sort 12 -
MB</searchString>
            <earliestTime>-24h@h</earliestTime>
            <latestTime>now</latestTime>
            <option
name="charting.axisTitleX.visibility">visible</option>
            <option
name="charting.axisTitleY.visibility">visible</option>
            <option name="charting.axisX.scale">linear</option>
            <option name="charting.axisY.scale">linear</option>
            <option name="charting.chart">pie</option>
            <option name="charting.chart.nullValueMode">gaps</option>
            <option
name="charting.chart.sliceCollapsingThreshold">0.01</option>
            <option name="charting.chart.stackMode">stacked</option>
            <option name="charting.chart.style">shiny</option>
            <option name="charting.drilldown">all</option>
            <option name="charting.layout.splitSeries">0</option>
            <option
name="charting.legend.labelStyle.overflowMode">ellipsisMiddle</option>
            <option name="charting.legend.placement">right</option>
          </chart>
        </row>
        <row>
          <table>
            <title>Detailed Traffic Past 24 Hours</title>
            <searchString>index=netflow | stats sum(bytes) as Bytes by
src_ip, dst_ip, dst_port | eval MB=Bytes/(1024*1024)| rename src_ip as
Source, dst_ip as Destination | table Source, Destination, MB | sort 25 -
MB</searchString>
            <earliestTime>-24h@h</earliestTime>
            <latestTime>now</latestTime>
          </table>
        </row>
</dashboard>
```



Detailed Traffic Past 24 Hours 33m ago

Source	Destination	MB
10.[redacted]	10[redacted]	2889.795936
10[redacted]	10[redacted]	2466.999004
10[redacted]	10.[redacted]	1753.980311
10.[redacted]	10.[redacted]	834.229033
10.[redacted]	10[redacted]	722.177040
[redacted]	66.[redacted]	515.715179
10.[redacted]	10.[redacted]	433.439330
93.1[redacted]	[redacted]	332.163364
10[redacted]	10[redacted]	292.543138
10[redacted]	10[redacted]	285.263664

« prev 1 2 3 next »

6.7. Additional queries and dashboards

The Git repository for this project contains additional queries and sample dashboards. To enrich the reports, lookup data specific to your environment can be added to identify specific networks, hosts, or services. Additionally, summary indexes can be created to speed up the processing of the reports that look at large data sets over time, such as the top talker or top protocol over past 24 hours reports. The features of Splunk are covered only in an introductory fashion, as the description of the full capabilities of the Splunk platform are beyond the scope of this project.

7. Conclusions

NetFlow data is a valuable tool for the security analyst. Coupled with the search capabilities and visualization options offered by Splunk, the use of NetFlow data for investigations or incident response can be utilized at a high level of effectiveness. The reference architecture described in this project is easy to deploy, but as is the case with many systems, the searches, systems of interest, essential services, and lookup data must be tuned to match the specifics of environment.

References

Chickowski, E. (2013, December 19). Using NetFlow Data For More Robust Network Security. Retrieved from http://www.darkreading.com/attacks-breaches/using-netflow-data-for-more-robust-network-security/d/d-id/1141085

Cisco IOS NetFlow - Cisco. (n.d.). Retrieved from http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

Claise, B., Trammell, B., & Aitken, P. (2013, September). RFC 7011. Retrieved from https://tools.ietf.org/html/rfc7011

Components of a Splunk Enterprise deployment. (n.d.). Retrieved from http://docs.splunk.com/Documentation/Splunk/6.2.1/Capacity/ComponentsofaSplunkEnterprisedeployment

Free vs. Enterprise | Splunk License Comparison Table | Splunk. (n.d.). Retrieved from http://www.splunk.com/view/SP-CAAAE8W

Install on Linux. (n.d.). Retrieved from http://docs.splunk.com/Documentation/Splunk/6.2.1/Installation/InstallonLinux

NetFlow || Version 5. (n.d.). Retrieved from https://www.plixer.com/support/netflow_v5.html

NetFlow version 9 - Cisco. (n.d.). Retrieved from http://www.cisco.com/c/en/us/products/ios-nx-os-software/netflow-version-9/index.html

Patterson, M. (2012, January 19). Threat detection with NetFlow: IP reputation - Mike Patterson. Retrieved from http://www.bradreese.com/blog/threat-protection-with-netflow.htm

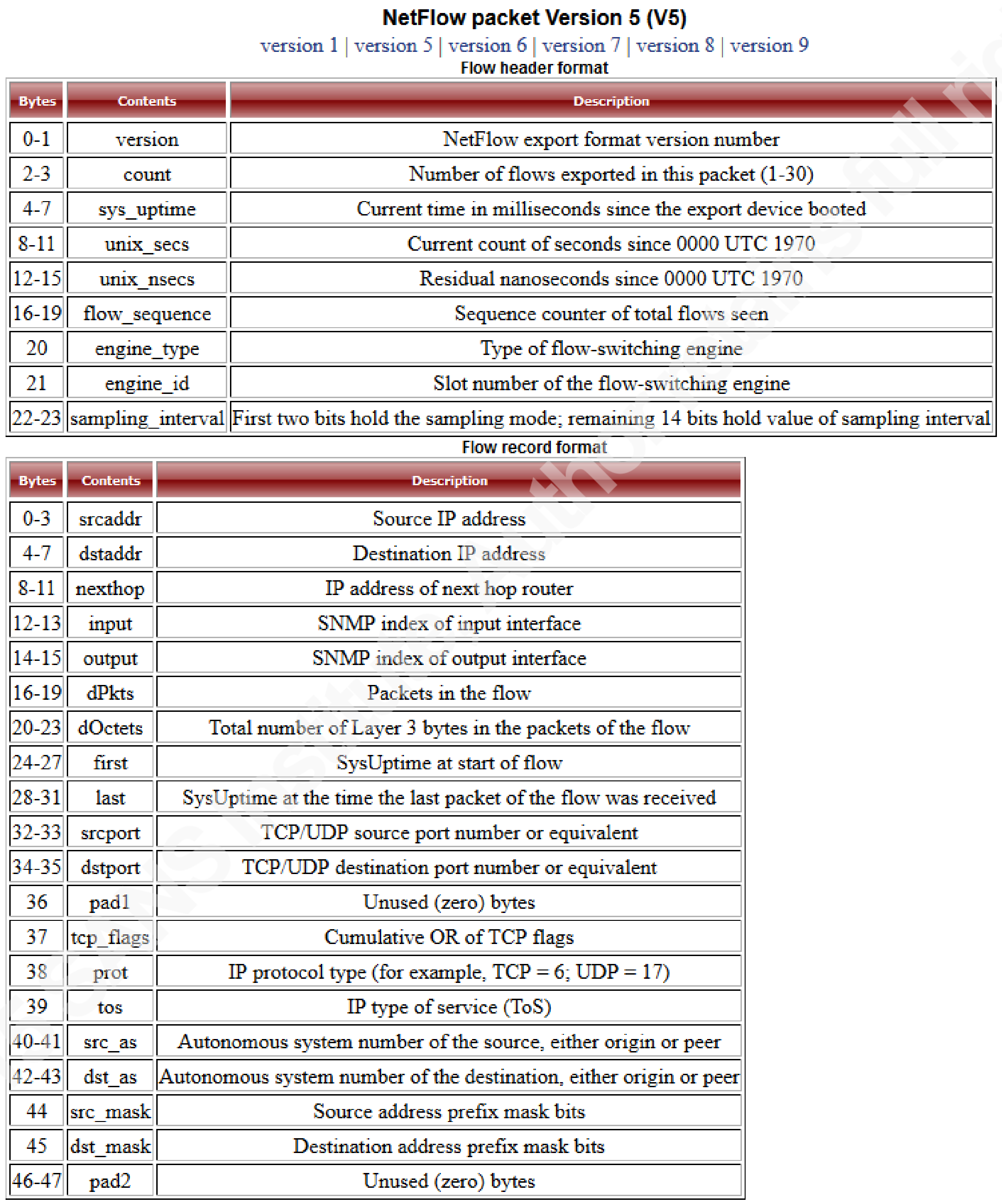
Patterson, M. (2012, January 13). NetFlow Behavior Analysis Systems : Limited Impact. Retrieved from http://blog.tmcnet.com/advanced-netflow-traffic-analysis/2012/01/netflow-behavior-analysis-systems-limited-impact.html

Patterson, M. (2013, March 22). NetFlow v5 Vs. NetFlow v9 | Extreme Networks. Retrieved from http://www.extremenetworks.com/netflow-v5-vs-netflow-v9

Scheck, M. (2009). Netflow For Incident Detection. Retrieved from http://www.first.org/global/practices/Netflow.pdf

David Mashburn, dmashburn@gmail.com

Appendix A – NetFlow version 5 format



NetFlow packet Version 5 (V5)

version 1 | version 5 | version 6 | version 7 | version 8 | version 9

Flow header format

Bytes	Contents	Description
0-1	version	NetFlow export format version number
2-3	count	Number of flows exported in this packet (1-30)
4-7	sys_uptime	Current time in milliseconds since the export device booted
8-11	unix_secs	Current count of seconds since 0000 UTC 1970
12-15	unix_nsecs	Residual nanoseconds since 0000 UTC 1970
16-19	flow_sequence	Sequence counter of total flows seen
20	engine_type	Type of flow-switching engine
21	engine_id	Slot number of the flow-switching engine
22-23	sampling_interval	First two bits hold the sampling mode; remaining 14 bits hold value of sampling interval

Flow record format

Bytes	Contents	Description
0-3	srcaddr	Source IP address
4-7	dstaddr	Destination IP address
8-11	nexthop	IP address of next hop router
12-13	input	SNMP index of input interface
14-15	output	SNMP index of output interface
16-19	dPkts	Packets in the flow
20-23	dOctets	Total number of Layer 3 bytes in the packets of the flow
24-27	first	SysUptime at start of flow
28-31	last	SysUptime at the time the last packet of the flow was received
32-33	srcport	TCP/UDP source port number or equivalent
34-35	dstport	TCP/UDP destination port number or equivalent
36	pad1	Unused (zero) bytes
37	tcp_flags	Cumulative OR of TCP flags
38	prot	IP protocol type (for example, TCP = 6; UDP = 17)
39	tos	IP type of service (ToS)
40-41	src_as	Autonomous system number of the source, either origin or peer
42-43	dst_as	Autonomous system number of the destination, either origin or peer
44	src_mask	Source address prefix mask bits
45	dst_mask	Destination address prefix mask bits
46-47	pad2	Unused (zero) bytes

Tables above adapted from Plixer.com ("NetFlow || Version 5," n.d.)

Appendix B – nfdump output format

The nfdump utility has multiple options for output format. The option selected for this project is the CSV output format. The command below shows the header for the fields that are exported by nfdump when the CSV output format is selected.

```
user@host:~$ sudo nfdump -o csv -r nfcapd.current

ts,te,td,sa,da,sp,dp,pr,flg,fwd,stos,ipkt,ibyt,opkt,obyt,in,out,sas,das,smk
,dmk,dtos,dir,nh,nhb,svln,dvln,ismc,odmc,idmc,osmc,mpls1,mpls2,mpls3,mpls4,
mpls5,mpls6,mpls7,mpls8,mpls9,mpls10,cl,sl,al,ra,eng,exid,tr

[flow data omitted]
```

Each of the flow fields are defined in the table below. This table is based on a review of the nfdump source code (refer to nf_common.c).

Abbreviation	**Data Field**
ts	Flow Start Time
te	Flow End Time
td	Flow duration
sa	Source IP address
da	Destination IP address
sp	Source Port
dp	Destination Port
pr	Protocol
ra flg	Flags
fwd	Forwarding Status
stos	Source Type of Service
ipkt	Input Packets
ibyt	Input Bytes
opkt	Output Packets
obyt	Output Bytes
in	Input Interface
out	Output Interface
sas	Source BGP autonomous system number
das	Destination BGP autonomous system number
smk	Source netmask
dmk	Destination netmask
dtos	Destination Type of Service
dir	Flow direction
nh	Next hop router
nhb	BGP next hop router
svln	Source VLAN
dvln	Destination VLAN
ismc	Input source MAC address
odmc	Output destination MAC address
idmc	Input destination MAC address
osmc	Output source MAC address
mpls1	MPLS label 1

David Mashburn, dmashburn@gmail.com

mpls2	MPLS label 2
mpls3	MPLS label 3
mpls4	MPLS label 4
mpls5	MPLS label 5
mpls6	MPLS label 6
mpls7	MPLS label 7
mpls8	MPLS label 8
mpls9	MPLS label 9
mpls10	MPLS label 10
cl	Client latency
sl	Server latency
al	Application latency
ra	Exporting system (router) IP address
eng	Engine type/ID
exid	Exporting system ID
tr	Flow received timestamp