Indexing Full Packet Capture Data With Flow

FloCon
January 2011

Randy Heins
Intelligence Systems Division
Overview

• Full packet capture systems can offer a valuable service provided that they are:
  – Retaining full fidelity data
  – Providing access to that data in a timely manner

• This discussion outlines lessons learned in developing a full packet capture system that meets these needs by using:
  – Abstracted flow representations
  – Application data extraction
  – Data indexing and caching

Goal: A full packet capture system capable of returning all relevant information quickly
Keeping Up With the Threats

- Know your threats
  - DOS
  - Data loss
  - Email phishing
  - Covert channels

- Know your sensors
  - What data is kept
  - How long can it be retained
  - How long it takes to retrieve

- Data is useless if it’s not actionable

The threats drive system design
Full Packet Capture Cycle

- **Capture Process Cycle**
  - Capture Data from Network
    - TCPdump, DaemonLogger
    - Rollover every X MB
    - Capture to RAMdisk for better performance
  - Analyze
    - Network and strings
    - Anomalies
  - Archive
    - Save data for future use
    - Pre-process certain types
  - Remove
    - Maintain retention standards
Volume Estimates

- Full packet capture of a saturated 1 Gbps link will yield:
  - 1 Day = 6TB
  - 1 Week = 42TB
  - 1 Month = 180TB

- Data is stored on sensors
  - Moving data to central storage would duplicate all traffic, not an option.
  - Data will be queried on sensors as well – causes disk I/O contention.
“Sensing” a Problem

• Indicators can be vague
  – “Anti-virus labs report a new malicious domain, www.badguy.com, has been used since December 15, 2010 to exploit vulnerable versions of web browsers.”

• My initial thoughts:
  1. Do I still have PCAP data from December 15?
     • Saving 1.5 months of full packet capture logs will be close to 45TB.
  2. Do I search for December 15 or the last 1.5 months?
     • Searching through 1 days worth of full packet capture logs using regular expressions on all port 80 data will take 6 hours. A query for 1.5 months will take 11+ days to complete.
  3. Should I filter on subject or URL?
     • Do both, because I don’t have an extra 11 days to wait for any subsequent queries.
Tiered Architecture

- Linear analysis of full packet capture files does not scale
  - Too much time is wasted searching for the needle in the haystack
  - File creation time is the only index provided by the capture, major inefficiency

- Possible Solution: A tiered schema to support analytical needs
  - High-fidelity data
  - Quick results using smart indices
  - Long data retention
Tier 1: Full Packet Capture

- Record all bytes captured off the wire using LibPCAP
  - TCPdump
  - DaemonLogger from Snort

- PCAP files are saved onto disk for analysis
  - TCPdump – rotates every X MB’s
  - DaemonLogger – can rotate by size or time interval
  - Filename useful if saved in format:
    - YYYY-MM-DD_HHMMSS.pcap

PCAP Archive

1 DAY OF FULL PACKET CAPTURE

- 1TB of disk space used
- 6 hours to query all data
Tier 1: Full Packet Capture – Use Case

Call For Data:
Identify traffic to www.badguy.com in the last 24 hours.

• Search PCAP files for regular expression:
  – /^Host: www.badguy.com/
  – Limit to port 80 for efficiency by use of BPF

• Effectiveness:
  – Accurate, low amount of false positives

• Cost:
  – Disk I/O: Reading 1TB (2,000 512MB files) of data may hinder other disk-bound applications, such as the capture process
  – Speed: Up to 6 hours for query to complete, not acceptable
Tier 2: NetFlow

- Flowmeter used to produce a Netflow representation of full packet capture data
  - SiLK YAF
  - softflowd

- Provides layer 4 summary*
  - *YAF applabel feature identifies some protocols

1 GB of disk used
1 minute to query
Tier 2: NetFlow – Use Case

Call For Data:
Identify traffic to www.badguy.com in the last 24 hours.

• Search NetFlow records:
  - --dip=[IP address of www.badguy.com]

• Effectiveness:
  - Low accuracy: traffic may be for another virtual host using the same IP
  - Limited context: protocol information is not given by NetFlow, this could be a non-HTTP process listening on port 80

• Cost:
  - Disk I/O: Reading 1GB of packed NetFlow is relatively low
  - Speed: Within several minutes for query to complete
Tier 3: AppFlow

• Looking for the best of both worlds:
  – The speed of NetFlow
  – The fidelity of full packet capture

• AppFlow - a hybrid approach:
  – Unique list of relevant attributes are extracted from each full packet capture file
  – Extract attributes that are the source of most queries:
    • SMTP - header elements, attachment filenames
    • HTTP – URI’s, user-agent strings, SSL certificate attributes
    • DNS – question/answer attributes
    • Layer 3 – source IP, destination IP
  – Context is provided by the associated full packet capture file

1 DAY OF APPFLOW

• 200MB of disk space used
• 4 seconds to query data
Tier 3: AppFlow

- Relevant attributes from each Full Packet Capture file are extracted into a corresponding AppFlow file

|---------------------|---------------------------|---------------------------|---------------------------|

|          | joe_smith@example.com  Meeting next week  www.example.com/ /files/document.pdf  host.example.com  Meeting_2011_01_24.doc  2015-10-22 05:00:00 | test.example.com  Fwd: Upcoming event  bob@example.com  Re: Wainscoting quote 10.132.53.21 /cgi-bin/temp/index.html | Fwd: Upcoming event  bob@example.com  Re: Wainscoting quote 10.132.53.21 /cgi-bin/temp/index.html  ftp.example.com  jnorthrop@example.com  /get_weather.php  test.example.com |

© 2011 Northrop Grumman Corporation
Tier 3: AppFlow – Use Case

Call For Data:
Identify traffic to www.badguy.com in the last 24 hours.

- **Search AppFlow records:**
  
  ```bash
  $ grep 'www.badguy.com' 2011-01-22*.appflow
  2011-01-22_034521.appflow: www.badguy.com
  2011-01-22_083200.appflow: www.badguy.com
  ```

- **Effectiveness:**
  - Decent Accuracy: ‘www.badguy.com’ may be part of an HTTP, SMTP, or DNS flow
  - No context: there is no association to the traffic

- **Cost:**
  - Disk I/O: Very low
  - Speed: Very fast
Data As An Index

- AppFlow serves as an efficient index for full packet capture files
  - Determine, “Is value X in the AppFlow index”?
    - Yes: then query the associated full packet capture file for related data
    - No: skip to the next file
  - Reduces disk I/O and query time by identifying the relevant full packet captures files

Query Complexity

- AppFlow: 513 MB, 57 KB, 52 KB, 52 KB, 61 KB, 56 KB
- Full PCAP: 512 MB, 512 MB, 512 MB, 512 MB, 512 MB
- Relevant Activity: 5 MB
More Efficient Indices - Bloom Filters

• Most analytical queries start with the question, “does this value exist in a set of data?”

• Bloom filters are specifically designed to answer that question\(^1\)
  – Great use-case presented by Chris Roblee in FloCon 2008 \(^2\)
  – Use a hashing algorithm to store a set of values
  – Returns a Boolean response to the existence of a value in a set
  – Can produce false positive but no false negatives
    • The probability of false negatives is tunable but more reliable Bloom filters increase the data structure size

• Easy to store AppFlow data in a Bloom filter
  – Convert file to Bloom filter in 14 lines of code
  – Store on disk as a serialized data structure

1 – Ripeanu & Lamnitchi - www.cs.uchicago.edu/~matei/PAPERS/bf.doc/
Bloom Filter Efficiency

- How well Bloom filters perform:
  - Sample: 1 day of full packet capture data
  - Query speed and storage efficiency drastically increase
  - The two operations complete in the same amount of time (6 hours):
    - Querying 1 day of full packet capture data
    - Querying 50+ years of AppFlow Bloom filters

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Size</th>
<th>Query Time</th>
<th>Time Speedup</th>
<th>Storage Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Packet Capture</td>
<td>1 TB</td>
<td>6 hours</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AppFlow</td>
<td>200 MB</td>
<td>4 seconds</td>
<td>5,400x</td>
<td>5,000x</td>
</tr>
<tr>
<td>AppFlow Bloom Filter</td>
<td>20 MB</td>
<td>1 second</td>
<td>21,600x</td>
<td>50,000x</td>
</tr>
</tbody>
</table>
Tier 4: Caching

• Bloom filters produce limited false positives
  – Associated full packet capture files must be queried to determine which are incorrect
  – That operation can be costly but is ultimately necessary with any index
  – Analyst clustering - the problem worsens when multiple users are conducting similar queries, each making the same mistakes

• Limit the amount of redundant queries for false positive results by caching the correct results in memory
  – memcached\(^1\) - an open-source distributed memory caching system
  – Distributed: values can be retrieved, set, or updated from remote systems
  – Values to store:
    • Paths to PCAP files with relevant information
    • Time range, BPF, and path to query result PCAP file

\(^1\) http://memcached.org
Tiers of Comparison

Data Storage Requirements For a Single Day

- **100KB**
- **200MB**
- **1GB**
- **1TB**

Levels:
- **Cache**
- **AppFlow**
- **NetFlow**
- **Full Packet Capture**
Conclusions

• Full packet is here to stay because the network will remain common to most incidents

• Attack vectors will change so tools need to remain flexible

• Indexing abstracted flow representations is one method for improving the gap between indicators and identification.