Detecting malicious traffic by using NetFlow data
By: Joey Dreijer, Student OS3
Gathering NetFlow data

- Router/Switch sends flow stats to external collector
- Collector receives and stores flow details
- Parser/interface reads flow from collector dump
NetFlow in short

- NetFlow data not just a 'term'
  - NetFlow (v9) specified in RFC3954
  - NetFlow commonly used from v5 and up

- NetFlow standardized to sent 'flow' characteristics
  - Stats such as bytes, packet number, port, session timer
  - Implemented in different (multi-vendor) routers/switches
  - Does not include packet content
  - *Request and response two different flows*
  - Often used for network performance measurement
Data required for research

- NetFlow collector stored the following details (using v5):
  - Source Address
  - Destination Address
  - Source Port
  - Destination Port
  - (TCP Flags)
  - Bytes send
  - Packets send
  - Time

<table>
<thead>
<tr>
<th>Date flow start</th>
<th>Duration Proto</th>
<th>Src IP Addr:Port</th>
<th>Dst IP Addr:Port</th>
<th>Packets</th>
<th>Bytes Flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-06-30 19:45:39.253</td>
<td>116.103 TCP</td>
<td>10.0.2.15:50494 -&gt; 62.69.166.15:80</td>
<td>46</td>
<td>6442</td>
<td>1</td>
</tr>
<tr>
<td>2014-06-30 19:45:39.253</td>
<td>116.103 TCP</td>
<td>62.69.166.15:80 -&gt; 10.0.2.15:50494</td>
<td>47</td>
<td>42669</td>
<td>1</td>
</tr>
<tr>
<td>2014-06-30 19:45:39.375</td>
<td>115.985 TCP</td>
<td>10.0.2.15:33675 -&gt; 74.125.136.94:80</td>
<td>8</td>
<td>1142</td>
<td>1</td>
</tr>
<tr>
<td>2014-06-30 19:45:39.375</td>
<td>115.985 TCP</td>
<td>74.125.136.94:80 -&gt; 10.0.2.1533675</td>
<td>7</td>
<td>640</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: NetFlow v5 is dinosaur old. Use v9 or IPFIX instead for more stats.
Data required for research

- Combining request/response to get the following data:
  - Source Address
  - Destination Address
  - Source Port
  - Destination Port
  - (TCP Flags)
  - Bytes Incoming
  - Bytes outgoing
  - Packets incoming
  - Packets outgoing
  - Average session time

<table>
<thead>
<tr>
<th>Date</th>
<th>Flow start</th>
<th>Duration</th>
<th>Proto</th>
<th>Src IP Addr:Port</th>
<th>Dst IP Addr:Port</th>
<th>Out Pkt</th>
<th>In Pkt</th>
<th>Out Byte</th>
<th>In Byte Flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-06-30</td>
<td>19:45:39.375</td>
<td>115.985</td>
<td>TCP</td>
<td>10.0.2.15:33675</td>
<td>74.125.136.94:80</td>
<td>7</td>
<td>8</td>
<td>640</td>
<td>1142</td>
</tr>
<tr>
<td>2014-06-30</td>
<td>19:45:39.396</td>
<td>115.961</td>
<td>TCP</td>
<td>10.0.2.15:46932</td>
<td>62.69.166.18:80</td>
<td>7</td>
<td>8</td>
<td>712</td>
<td>1517</td>
</tr>
</tbody>
</table>
Collecting NetFlow data

- SoftFlowd sends NetFlow data to collector (nfcapd). Optional: Pcap or Interface as input
- NetFlow data stored in binary format
- Format parsed by Python wrapper and nfdump (custom patched pynfdump_altered)
Initial protocol analysis

- Gathering 'known-good' traffic
- Generating 'known-bad' traffic
  - Comparing differences / similarities
  - Storing usefull comparison data

For each:
- Dst. Port
- In Bytes
- Out Bytes
- In Packets
- Out Packets
- Avg Time

Database containing:
- Max/min values
- Averages
- Standard Deviation
Comparing NetFlow data

- Traffic analysis; comparing 'real-time' binary (nfdump) vs stored (MySQL)
- 'Anomaly detection' based on selected metrics/profile
- Maximum range via standard deviation
  - **Note:** Only *if* possible. Not all traffic can be normalized

![Diagram of NetFlow Analysis System]

- **Metrics** → **Database**
- **Statistics** → **Database**
- **Analyzer** → **NetFlow Dump**
Detecting Tunnels / Covert Channels

- **Example 1:** DNS Tunnels
- DNS may have 'normal behaviour'
- Tunneling via DNS abnormal statistics based on metric x?
- Verify differentiation per metric

![Graph showing 'Starting' DNS Tunnel. Not sending data yet.]

Compared to +- 2 million DNS Flows
Detecting Tunnels / Covert Channels

- Previous examples done via anomaly detection
- Known-good database used as reference
- Pre-defined profile (ie. alert only if packets and time mismatch by x)

\[
\text{anomaly} = (\text{max difference} \times \text{standard deviation}) + \text{average}
\]

If anomaly is larger than current flow:
If packetAnomaly and timeAnomaly:
Generate Alert
Detecting Tunnels / Covert Channels

- Why are multiple metrics important? (and/and policy)
- NetFlow parser shows incorrect flows with much traffic
- True automated anomaly detection shows many FP's

**Example:**
10.10.0.2:50001 → 8.8.8.8:53
Packets: 4, time: 4001 seconds (….?)

- Actually 2 DNS requests on different times
  - However, identical source port and destination lets 'nfdump' think it is the same flow → results in False Positive
Detecting Tunnels / Covert Channels

- Comparing with realistic dataset
- 17 million flows from GuestNet
  - Literal flow dump, can contain 'malicious' flows
  - Both bad and good traffic?

- 2 million DNS responses
  - Results in 0.0005% hits based on combined metrics
    - Includes previous 'bug' with multiple sessions combined due to identical ports and destinations
    - Uncertain if actual tunnels inside dump
Other uses

- **Example 2: NMAP Scan**
- Aggregated NetFlow shows requests and response
- NetFlow shows flow with no responses for filtered ports

- Probability 'x' amount of ports do not reply within 'y' amount of time based on 'z' amount of retries/packets
Other uses

- Small problem with portscans....

- Nfcapd holds a default 5 minute NetFlow cache
- Not all flows stored after cache timer
  - Waits for finished sessions before storing flow
  - Half open TCP sessions will be cached until timeout
  - Timeout can last 20 minutes depending on config
NetFlow Analysis: Detecting covert channels on the network

Introduction

Research

Tooling

Detection

Demo

Conclusion

DEMO
Conclusion

- NetFlow only sends limited amount of information
  - Does not say anything about packet contents

- Fairly easy to detect 'well-know' and publicly available tunnels and scans

- Covert Channels / tunnels always possible; attacker has all the time in the world.
  - Craft pingtunnel to send fixed size packets every second to conform the 'default' behaviour.