Network Troubleshooting Using ntopng
Luca Deri <deri@ntop.org>
Outlook

• Part 1: Introduction to ntopng
  ◦ ntopng architecture and design.
  ◦ ntopng as a flow collector.
  ◦ Exploring system activities using ntopng.

• Part 2: ntopng+Wireshark Monitoring Use Cases
  ◦ Using ntopng.
  ◦ ntopng and Wireshark.
  ◦ Advanced monitoring with ntopng.
  ◦ Future roadmap items.
About ntop.org

• ntop develops open source network traffic monitoring applications.

• ntop (circa 1998) is the first app we released and it is a web-based network monitoring application.

• Today our products range from traffic monitoring, to high-speed packet processing, deep-packet inspection, and IDS/IPS acceleration (snort, Bro and suricata).
ntop’s Approach to Traffic Monitoring

- Ability to capture, process and (optionally) transmit traffic at line rate, any packet size.
- Leverage on modern multi-core/NUMA architectures in order to promote scalability.
- Use commodity hardware for producing affordable, long-living (no vendor lock), scalable (use new hardware by the time it is becoming available) monitoring solutions.
- Use open-source to spread the software, and let the community test it on unchartered places.
Some History

• In 1998, the original ntop has been created.

• It was a C-based app embedding a web server able to capture traffic and analyse it.

• Contrary to many tools available at that time, ntop used a web GUI to report traffic activities.

• It is available for Unix and Windows under GPL.
ntop Architecture

HTTP/HTTPS ↔ RRD ↔ Cisco NetFlow
InMon sFlow

<table>
<thead>
<tr>
<th>Report Engine</th>
<th>Plugins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet Analyser</td>
<td>Traffic Rules</td>
</tr>
<tr>
<td>Packet Sniffer</td>
<td></td>
</tr>
</tbody>
</table>

ntop for Win
Top Senders
Host  Total
192.168.104 991.4

Top Receivers
Sent  312.5kB

Options
Why was ntop obsolete?

• Its original LAN-oriented design prevented ntop from handling more than a few hundred Mbit.
• The GUI was an old (no fancy HTML 5) monolithic piece written in C so changing/ extending a page required a programmer.
• ntop could not be used as web-less monitoring engine to be integrated with other apps.
• Many components were designed in 1998, and it was time to start over (spaghetti code).
ntopng Design Goals

• Clean separation between the monitoring engine and the reporting facilities.
• Robust, crash-free engine (ntop was not really so).
• Platform scriptability for enabling extensions or changes at runtime without restart.
• Realtime: most monitoring tools aggregate data (5 mins usually) and present it when it’s too late.
• Many new features including HTML 5-based dynamic GUI, categorisation, DPI.
ntopng Architecture

- Three different and self-contained components, communicating with clean API calls.

- Users
  - HTTP
  - Lua-based Web Reports
    - Lua API Calls
  - nDPI-based C++ Monitoring Engine
    - PF_RING C API Calls
  - PF_RING Kernel Module and Drivers
  - (Linux) Kernel
  - Internet Traffic
  - Data Cache

SharkFest 2015 - Computer History Museum June 22-15, 2015
ntopng Monitoring Engine

- Coded in C++ and based on the concept of flow (set of packets with the same 6-tuple).
- Flows are inspected with a home-grown DPI-library named nDPI aiming to discover the “real” application protocol (no ports are used).
- Information is clustered per:
  - (Capture) Network Device
  - Flow
  - Host
  - High-level Aggregations
Local vs Remote Hosts [1/2]

• ntopng keeps information in memory at different level of accuracy in order to save resources for hosts that are not “too relevant”.
• For this reason at startup hosts are divided in:
  ◦ Local hosts/System Host
    The local host where ntopng is running as well the hosts belonging to some “privileged” IPv4/v6 networks. These hosts are very relevant and thus ntopng keeps full statistics.
  ◦ Remote hosts
    Non-local hosts for which we keep a minimum level of detail.
Local vs Remote Hosts [2/2]

- For local hosts (unless disabled via preferences) are kept all L7 protocol statistics, as well as basic statistics (e.g. bytes/packets in/out).
- No persistent statistics are saved on disk.
- A system host is the host where ntopng is running and it is automatically considered local as well the networks of its ethernet interfaces.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ASN</td>
<td>2597 [ Registry of ccTLD it - IIT-CNR ]</td>
</tr>
<tr>
<td>Name</td>
<td>pc-deri.nic.it [ Local ] [ System ]</td>
</tr>
</tbody>
</table>
Information Lifecycle

- ntopng keeps in memory live information such as flows and hosts statistics.
- As the memory cannot be infinite, periodically non-recent information is harvested.
- Users can specify preferences for data retention:

![Data Purge Screen](image.png)
Packet Processing Journey

1. Packet capture: PF_RING, netfilter (Linux) or libpcap.
2. Packet decoding: no IP traffic is accounted.
3. IPv4/v6 Traffic only:
   1. Map the packet to a 6-tuple flow and increment stats.
   2. Identify source/destination hosts and increment stats.
   3. Use nDPI to identify the flow application protocol
      1. UDP flows are identified in no more than 2 packets.
      2. TCP Flows can be identified in up to 15 packets in total, otherwise the flow is marked as “Unknown”.
4. Move to the next packet.
PF_RING

- In 2004 we have realised the Linux kernel was not efficient enough to fulfil our packet capture requirements and thus we have written a in-kernel circular buffer named PF_RING.
Moving towards 10 Gbit and above

- The original PF_RING is a good solution up to 3/5 Gbit but not above as the cost of packet copy into the ring is overkilling.
- PF_RING ZC (Zero Copy) is an extension that allows packets to received/transmitted in zero copy similar to what FPGA-accelerated cards (e.g. Napatech and Accolade) do in hardware.
PF_RING (ZC) and ntopng

Using PF_RING (ZC) with ntopng has several benefits:

- ntopng can scale to 10 Gbit and above by spawning several ntopng instances each bound to a (few) core(s).
- It is possible to send the same packet to multiple apps. For instance it is possible to send the same packet to ntopng (for accounting purposes) and n2disk (ntop’s application for dumping packet-to-disk at multi-10G) and/or and IDS (e.g. Suricata and snort).
The need for DPI in Monitoring [1/2]

- Limit traffic analysis at packet header level it is no longer enough (nor cool).
- Network administrators want to know the real protocol without relying on the port being used.
- Selected protocols can be “precisely dissected” (e.g. HTTP) in order to extract information, but on the rest of the traffic it is necessary to tell network administrators what is the protocol flowing in their network.
The need for DPI in Monitoring [2/2]

• DPI (Deep Packet Inspection) is a technique for inspecting the packet payload for the purpose of extracting metadata (e.g. protocol).

• There are many DPI toolkits available but they are not what we looked for as:
  ◦ They are proprietary (you need to sign an NDA to use them), and costly for both purchase and maintenance.
  ◦ Adding a new protocol requires vendor support (i.e. it has a high cost and might need time until the vendor supports it) = you’re locked-in.

• On a nutshell DPI is a requirement but the market does not offer an alternative for open-source.
Say hello to nDPI

- ntop has decided to develop its own LGPLv3 DPI toolkit in order to build an open DPI layer for ntop and third party applications.
- Supported protocols (> 180) include:
  - P2P (Skype, BitTorrent)
  - Messaging (Viber, Whatsapp, MSN, The Facebook)
  - Multimedia (YouTube, Last.gm, iTunes)
  - Conferencing (Webex, CitrixOnLine)
  - Streaming (Zattoo, Icecast, Shoutcast, Netflix)
  - Business (VNC, RDP, Citrix, *SQL)
nDPI Overview

• Portable C library (Win and Unix, 32/64 bit).
• Designed for user and kernel space
  ◦ Linux ndpi-netfilter implements L7 kernel filters
• Used by many non-ntop projects (e.g. xplico.org) and part of Linux distributions (e.g. Debian).
• Able to operate on both plain ethernet traffic and encapsulated (e.g. GTP, GRE...).
• Ability to specify at runtime custom protocols (port or hostname - dns, http, https -based).
nDPI on ntopng

• In ntopng all flows are analysed through nDPI to associate an application protocol to them.
• L7 statistics are available per flow, host, and interface (from which monitoring data is received).
• For network interfaces and local hosts, nDPI statistics are saved persistently to disk (in RRD format).
nDPI Protocol Clustering

nDPI can cluster protocols into categories:

- Safe (e.g. SSH)
- Acceptable (e.g. HTTP)
- Fun (e.g. YouTube)
- Unsafe (e.g. POP3)
- Potentially dangerous (e.g. Tor)
- Unrated (e.g. Unknown Protocol)
nDPI on ntopng: Interface Report [1/2]
nDPI on ntopng: Interface Report [2/2]
ntopng and Redis

• Redis is an open source key-value in-memory database.

• ntop uses it to cache data such as:
  ◦ Configuration and user preferences information.
  ◦ DNS name resolution (numeric to symbolic).
  ◦ Volatile monitoring data (e.g. hosts JSON representation).

• Some information is persistent (e.g. preferences) and some is volatile: ntopng can tell redis how long a given value must be kept in cache.
Lua-based ntopng Scriptability [1/3]

• A design principle of ntopng has been the clean separation of the GUI from the engine (in ntop it was all mixed).
• This means that ntopng can (also) be used (via HTTP) to feed data into third party apps such as Nagios or OpenNMS.
• All data export from the engine happens via Lua similar to what happens in Wireshark.
• Lua methods invoke the ntopng C++ API in order to interact with the monitoring engine.
Lua-based ntopng Scriptability [2/3]

- /scripts/callback/ scripts are executed periodically to perform specific actions.
- /scripts/lua/ scripts are executed only by the web GUI.
- Example: http://ntopng:3000/lua/flow_stats.lua
Lua-based ntopng Scriptability [3/3]

• ntopng defines (in C++) two Lua classes:
  ◦ interface
    • Hook to objects that describe flows and hosts.
    • Access to live monitoring data.
  ◦ ntop
    • General functions used to interact with ntopng configuration.

• Lua objects are usually in “read-only” mode
  ◦ C++ sets their data, Lua reads data (e.g. host.name).
  ◦ Some Lua methods (e.g. interface.restoreHost()) can however modify the information stored in the engine.
ntopng as a NetFlow/sFlow Collector [1/3]

• The “old” ntop included a NetFlow/sFlow collector. Considered the effort required to support all the various NetFlow dialects (e.g. Cisco ASA flows are not “really” flows), in ntopng we have made a different design choice.
ntopng as a NetFlow/sFlow Collector [2/3]

• nProbe (a home-grown NetFlow/sFlow collector/probe) is responsible for collecting/generating flows and convert them to JSON so that ntopng can understand it.

• The communication ntopng <-> nProbe is over ØMQ a simple/fast messaging system that allows the two peers to be decoupled while:
  ◦ Avoiding “fat” communication protocols such as HTTP.
  ◦ Relying on a system that works per message (no per packet) and handles automatic reconnection if necessary.
ntopng as a NetFlow/sFlow Collector [3/3]

Flows are sent in the following format


• Where:
  ○ “<Element ID>”: <value> (example 8 = IPV4_SRC_ADDR)

• Contrary to what happens in NetFlow/sFlow
  ntopng (collector) connects to nProbe (probe) and
  fetches the emitted flows. Multiple collectors can
  connect to the same probe. No traffic is created
  when no collector is attached to the probe.
Flow Collection Setup: an Example

Flow collection/generation (nProbe)

• Probe mode
  nprobe --zmq "tcp://*:5556" -i eth1 -n none

• sFlow/NetFlow collector mode
  nprobe --zmq "tcp://*:5556" -i none -n none --collector-port 2055

Data Collector (ntopng)

• ntopng -i tcp://127.0.0.1:5556
Creating ntopng Clusters [1/2]

• ntopng is not only a flow collector, but it can export flows in the same JSON format used in the received flows.
• This allows complex clusters to be created:
Creating ntopng Clusters [2/2]

- In many companies, there are many satellite offices and a few central aggregation points.
- Using ØMQ (both ntopng and nProbe flows are in the same format) it is possible to create a hierarchy of instances.
- Each node aggregates the traffic for the instances “below” it, so that at each tree layer you have a summarised view of the network activities.
System+Network Monitoring [1/3]

This is how most system management tools work on Linux:

1. `ls/of` accesses the `/proc` directory.
2. `/proc` contains various files and directories for system monitoring.
3. Tools like `top`, `iostat`, `blktrace`, and `dtrace` are used to monitor system performance and activity.
System+Network Monitoring [2/3]

- Using ntopng/nProbe you can see the flows that are being exchanged across systems but it is not possible to know more than that.
System+Network Monitoring [3/3]

• It would be desirable to know exactly what is the process originating the traffic observed and what resources the process is using while generating such traffic.

• In essence we would like to see this picture:
Welcome to Sysdig

• Sysdig is a Linux framework developed by Sysdig Cloud for capturing system calls.
• The kernel module intercepts the calls.
• The user-space libs receive and interpret the received calls.
ntopng+nProbe+sysdig [1/2]

• In order to activate system+network monitoring, it is necessary to load the sysdig kernel module and start nProbe (flow probe) as follows:

  nprobe -T "%IPV4_SRC_ADDR %L4_SRC_PORT %IPV4_DST_ADDR %L4_DST_PORT %IN_PKTS %IN_BYTES %FIRST_SWITCHED %LAST_SWITCHED" %TCP_FLAGS %PROTOCOL @PROCESS% %L7_PROTO
  --zmq "tcp://*:1234" -i any --dont-drop-privileges -t 5 -b 2

• Then start ntopng (flow collector) as follows:

ntopng+nProbe+sysdig [2/2]

• When ntopng receives flow enriched with system information, it interprets it, and depicts:
  ◦ The process-to-flow association.
  ◦ For flows whose peers are hosts monitored by nProbe instances, it “glues” the flows together.
  ◦ The process call father/process hierarchy is depicted.
  ◦ The overall system process view including the process relationships.
Process Network Communications
Flow/Process Drill-down [1/2]
# Flow/Process Drill-down [2/2]

## Client Process Information

<table>
<thead>
<tr>
<th>User Name</th>
<th>deri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process PID/Name</td>
<td>13058/ntopng [son of 11235/tcsh]</td>
</tr>
<tr>
<td>Average CPU Load</td>
<td>0.71 %</td>
</tr>
<tr>
<td>I/O Wait Time Percentage</td>
<td>0 %</td>
</tr>
<tr>
<td>Memory Actual / Peak</td>
<td>1.4 MB / 1.46 MB [95.7%]</td>
</tr>
<tr>
<td>VM Page Faults</td>
<td>0</td>
</tr>
</tbody>
</table>

## Server Process Information

<table>
<thead>
<tr>
<th>User Name</th>
<th>redis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process PID/Name</td>
<td>1769/redis-server [son of 1/init]</td>
</tr>
<tr>
<td>Average CPU Load</td>
<td>0.12 %</td>
</tr>
<tr>
<td>I/O Wait Time Percentage</td>
<td>0 %</td>
</tr>
<tr>
<td>Memory Actual / Peak</td>
<td>344.13 KB / 344.13 KB [100%]</td>
</tr>
<tr>
<td>VM Page Faults</td>
<td>0</td>
</tr>
</tbody>
</table>

**Flow-to-Process binding**

**Dynamically Updated**
ntopng and Big Data [1/2]

• Using SQLite to save flows persistently is good when flows are not too many and the system that runs ntopng has storage.
• For large deployments or disk-less systems (e.g. ARM-based PCs) it is desirable to upload flows on remote, cloud-based, systems able to scale with the number of flows.
• In essence ntopng has been opened to what is currently defined as “big data” systems that can scale with data in volume and speed.
ntopng and Big Data [2/2]

• You can configure ntopng to export flow data directly into ElasticSearch and display them with Kibana

```bash
ntopng -F "es;flows;ntopng-%Y.%m.%d;http://XYZ:9200/_bulk;" -i eth1
```
ntopng Kibana Dashboard [1/2]
ntopng Kibana Dashboard [2/2]

- The GUI refreshes automatically as new data arrive and users can drill down data or visualise raw flows.
ntopng on Virtual Environments

• ntopng has been packaged for major Linux distributions such as Debian/Ubuntu, CentOS/RedHat and also FreeBSD and OSX (brew): installation couldn’t be simpler.
• However the current trend is going towards virtualised environments (not just VMs such as VMware) and IaaS (Infrastructure as a Service) and thus we need to support them.
Using ntopng To Enforce Policies [1/2]

• With ntopng 2.0 it is possible not just to monitor traffic but also to enforce network policies.
• In this case, ntopng works as an inline device operating as a network bridge.

ntopng
(Bump-In-The-Wire)
Using ntopng To Enforce Policies [2/2]

• In inline mode ntopng can be configured to let specific traffic pass/not-pass (i.e. drop all Skype traffic) or shape.
Embedding ntopng [1/3]

- Historically we have started our first embed attempt in 2003 with the Cyclades TS100.
- The nBox was used to analyse traffic then sent to ntop for representation.
- After 10 years we have tried again with ntopng.
Embedding ntopng [2/3]

• It is a while that we are working towards a cheap platform for everyone…
Embedding ntopng [3/3]

• It is also possible to combine a small ARM device with a copper network tap using a CatchWire device.

• Or for those who need more horsepower, PCEngines can offer you a cheap x64 device to run ntopng on.
ntopng and Wireshark: Real Life Monitoring Use Cases
Using ntopng with Wireshark

• Wireshark has been traditionally used for in-depth packet analysis.
• Usually Wireshark cannot be used as a long-term, permanent monitoring tool, but rather as tool used to analyse specific issues.
• **Combining** ntopng with Wireshark can enable you to implement **permanent monitoring** while being able to analyse in detail specific packets. In essence: the best of both worlds.
Analysing HTTP Traffic [1/4]

• Wireshark allows you to follow TCP streams and analyse their content.
Analysing HTTP Traffic [2/4]

• It is also possible to analyse sessions more in detail...
Analysing HTTP Traffic [3/4]

- ntopng allows people to visualise flows in a realtime list
Analysing HTTP Traffic [4/4]

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow Peers</td>
<td>lucas-imac.homenet.telecomitalia.it:84578</td>
</tr>
<tr>
<td>Protocol</td>
<td>TCP / HTTP</td>
</tr>
<tr>
<td>First / Last Seen</td>
<td>15/03/2015 18:58:19 [19 sec ago] 15/03/2015 18:58:35 [3 sec ago]</td>
</tr>
<tr>
<td>Total Traffic Volume</td>
<td>313.43 KB</td>
</tr>
<tr>
<td>Client vs Server Traffic Breakdown</td>
<td>83.184.221.133:80</td>
</tr>
<tr>
<td>Network Latency Breakdown</td>
<td>24.091 ms (server)</td>
</tr>
<tr>
<td>Client to Server Traffic</td>
<td>155 Pkts / 8.62 KB</td>
</tr>
<tr>
<td>Server to Client Traffic</td>
<td>214 Pkts / 304.82 KB</td>
</tr>
<tr>
<td>TCP Flags</td>
<td>SYN, PUSH, ACK - This flow is active.</td>
</tr>
<tr>
<td>Actual / Peak Throughput</td>
<td>182.24 bps ➙ / 240.89 Kbit/s</td>
</tr>
<tr>
<td>HTTP Method</td>
<td>GET</td>
</tr>
<tr>
<td>Server Name</td>
<td>download.cdn.mozilla.net</td>
</tr>
<tr>
<td>URL</td>
<td>/pub/firefox/releases/36.0.1/update/mac/en-US/firefox-35.0.1-36.0.1.partial.mar</td>
</tr>
<tr>
<td>Response Code</td>
<td>206</td>
</tr>
</tbody>
</table>
Network Health Analysis [1/3]

• Wireshark allows you to analyse packet/connection issues, including:
  ◦ Retransmissions
  ◦ Packets Out-Of-Order
  ◦ Packets Lost
Network Health Analysis [2/3]

Active Flows

Flows to Pay Attention

<table>
<thead>
<tr>
<th>Application</th>
<th>L4 Proto</th>
<th>Client</th>
<th>Server</th>
<th>Duration</th>
<th>Breakdown</th>
<th>Actual Thpt</th>
<th>Total Bytes</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>UDP</td>
<td>lucas-imac.homenet...</td>
<td>192.168.1.255:57621</td>
<td>1 h, 10 min, 38 sec</td>
<td>Client</td>
<td>0 bps</td>
<td>12.01 KB</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>UDP</td>
<td>lucas-imac.homenet...</td>
<td>broadcasthost:17500</td>
<td>1 h, 11 min, 8 sec</td>
<td>Client</td>
<td>0 bps</td>
<td>96.36 KB</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>UDP</td>
<td>lucas-imac.homenet...</td>
<td>192.168.1.255:17500</td>
<td>1 h, 11 min, 8 sec</td>
<td>Client</td>
<td>0 bps</td>
<td>96.36 KB</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>TCP</td>
<td>pc-der.nic.it :2222</td>
<td>lucas-imac.homenet...</td>
<td>1 h, 10 min, 52 sec</td>
<td>Client/Server</td>
<td>0 bps</td>
<td>126.98 KB</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>TCP</td>
<td>lucas-imac.homenet...</td>
<td>ash-ra1-3a.sjc.dropb... :80</td>
<td>1 h, 9 min, 57 sec</td>
<td>Client/Server</td>
<td>0 bps</td>
<td>170.26 KB</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>TCP</td>
<td>pc-der.nic.it :2222</td>
<td>lucas-imac.homenet...</td>
<td>1 h, 10 min, 40 sec</td>
<td>Client/Server</td>
<td>0 bps</td>
<td>34.44 KB</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>TCP</td>
<td>lucas-imac.homenet...</td>
<td>pc-der.nic.it :2222</td>
<td>36 min, 28 sec</td>
<td>Client/Server</td>
<td>0 bps</td>
<td>2.26 MB</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>TCP</td>
<td>lucas-imac.homenet...</td>
<td>net-93-64-151-231.cu.. :2220</td>
<td>22 min, 13 sec</td>
<td>Client/Server</td>
<td>21.67 Kbit/s</td>
<td>1.58 MB</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>TCP</td>
<td>lucas-imac.homenet...</td>
<td>ion3-accesspoint-a10... :4070</td>
<td>2 min, 1 sec</td>
<td>Client/Server</td>
<td>0 bps</td>
<td>643.01 KB</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>TCP</td>
<td>lucas-imac.homenet...</td>
<td>webmail3.ii.tcn.it :993</td>
<td>49 sec</td>
<td>Client/Server</td>
<td>1.54 Kbit/s</td>
<td>10.77 KB</td>
<td></td>
</tr>
</tbody>
</table>

Showing 1 to 10 of 28 rows
Network Health Analysis [3/3]

<table>
<thead>
<tr>
<th>(Router) MAC Address</th>
<th>D0:D4:12:C6:73:F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address</td>
<td>194.132.198.242</td>
</tr>
<tr>
<td>ASN</td>
<td>Spotify Technology SARL [ ASN 43650 ]</td>
</tr>
<tr>
<td>Name</td>
<td>lon3-accesspoint-a10.lon3.spotify.com</td>
</tr>
<tr>
<td></td>
<td>194.132.198.242</td>
</tr>
<tr>
<td>First / Last Seen</td>
<td>15/03/2015 19:10:02 [1 min, 39 sec ago]</td>
</tr>
<tr>
<td></td>
<td>15/03/2015 19:11:15 [26 sec ago]</td>
</tr>
<tr>
<td>Sent vs Received Traffic Breakdown</td>
<td></td>
</tr>
<tr>
<td>Traffic Sent / Received</td>
<td>433 Pkts / 537.55 KB</td>
</tr>
<tr>
<td>Flows Active / Total</td>
<td>'As Client'</td>
</tr>
<tr>
<td></td>
<td>0 / 0</td>
</tr>
<tr>
<td>TCP Packets Sent Analysis</td>
<td></td>
</tr>
<tr>
<td>Retransmissions</td>
<td>0 Pkts</td>
</tr>
<tr>
<td>Out of Order</td>
<td>40 Pkts</td>
</tr>
<tr>
<td>Lost</td>
<td>20 Pkts</td>
</tr>
</tbody>
</table>
Network Performance Analysis [1/4]

- With TCP, it is possible to measure the network latency by analysing the 3-way handshake

Observation Point

- Wireshark allows you to analyse packets delays. The idea is to make this simpler to read to everyone.
### Network Performance Analysis [3/4]

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>TCP / Google</td>
</tr>
<tr>
<td>First / Last Seen</td>
<td>09/02/2007 11:54:32 [8 years, 37 days, 10 h, 40 sec ago]</td>
</tr>
<tr>
<td>Total Traffic Volume</td>
<td>42.79 KB</td>
</tr>
<tr>
<td>Network Latency Breakdown</td>
<td>1.14 ms (client) - 27.290 ms (server)</td>
</tr>
<tr>
<td>Client to Server Traffic</td>
<td>20 Pkts / 1.93 KB</td>
</tr>
<tr>
<td>Server to Client Traffic</td>
<td>32 Pkts / 40.85 KB</td>
</tr>
<tr>
<td>TCP Flags</td>
<td>SYN RST FLUSH ACK</td>
</tr>
<tr>
<td>Actual / Peak Throughput</td>
<td>0 bps / 0 bps</td>
</tr>
<tr>
<td>Server Name</td>
<td>m1.2mdn.net</td>
</tr>
</tbody>
</table>

Do you finally know where is the higher latency: client or server?

- Similar to network latency, it is possible to compute the service response time.
- It can be computed for selected protocols (e.g. HTTP) with request/reply behaviour.
Download or Upload?

- In Wireshark it is not possible to mark packet directions and thus to easily understand the relevant packet direction.

Current Network Load

Download or Upload?
Packets Never Lie [1/3]

• Suppose that for specific hosts (e.g. for which an IDS has reported security issues) you want to save raw packets.
• Suppose that your host is under attack or is attacking somebody (e.g. portscan).
• Suppose that you want to save packets of unknown (i.e. not detected by nDPI) communications for inspection (or for improving nDPI).
• …then you need raw packets (pcap).
Packets Never Lie [2/3]

Only Under Attack

Lightweight Packet-to-Disk

<table>
<thead>
<tr>
<th>Packet Dump To Disk</th>
<th>Dump Traffic To Disk</th>
</tr>
</thead>
</table>
| Packet Dump To Tap  | Dump Traffic To Tap (tap0) |}

- **Sampling Rate**
  - 1:1
  - NOTE: Sampling rate is applied only when dumping packets caused by a security alert (e.g., a volumetric DDoS attack) and not to those hosts/flows that have been marked explicitly for dump.

- **Dump To Disk Parameters**
  - **Max Packets per File**
    - 10000
    - Maximum number of packets to store on a pcap file before creating a new file.
  - **Max Duration of File**
    - 300 sec
    - Maximum pcap file duration before creating a new file.
    - NOTE: A dump file is closed when it reaches the maximum size or duration specified.
Packets Never Lie [3/3]

- You can now see in realtime what is happening inside ntopng at packet level…
- …and at the same time ntopng generates pcap files for you.
Triggering Alerts [1/3]

• In Wireshark it is possible to identify issues and colour packets accordingly.

• In ntopng it is possible to analyse traffic and trigger alerts when specific conditions happen. Example host X has made more than Y bytes of peer-to-peer traffic.

• ntopng allows network administrators to set threshold for triggering alerts.
Triggering Alerts [2/3]

Per-host Alert Preferences

Observation Period

Condition

Thresholds
### Triggering Alerts [3/3]

#### Flow Alert (No Threshold)

<table>
<thead>
<tr>
<th>Action</th>
<th>Date</th>
<th>Severity</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sun Mar 15 20:52:20 2015</td>
<td>Error</td>
<td>TCP SYN Flood</td>
<td>Host 192.168.1.92 is a SYN flooder [68 SYNs sent in the last 3 sec] TCP 192.168.1.92:60970 &gt; 23.50.145.215:443 [proto: 0/Unknown][1/0 pkts][78/0 bytes]</td>
</tr>
<tr>
<td></td>
<td>Sun Mar 15 20:50:05 2015</td>
<td>Error</td>
<td>TCP SYN Flood</td>
<td>Host 192.168.1.92 is a SYN flooder [72 SYNs sent in the last 3 sec] TCP 192.168.1.92:60321 &gt; 37.252.170.95:80 [proto: 0/Unknown][1/0 pkts][78/0 bytes]</td>
</tr>
<tr>
<td></td>
<td>Sun Mar 15 20:49:46 2015</td>
<td>Error</td>
<td>TCP SYN Flood</td>
<td>Host 192.168.1.92 is a SYN flooder [64 SYNs sent in the last 3 sec] TCP 192.168.1.92:60172 &gt; 64.202.112.8:60 [proto: 0/Unknown][1/0 pkts][78/0 bytes]</td>
</tr>
<tr>
<td></td>
<td>Sun Mar 15 20:49:24 2015</td>
<td>Error</td>
<td>TCP SYN Flood</td>
<td>Host 192.168.1.92 is a SYN flooder [65 SYNs sent in the last 3 sec] TCP 192.168.1.92:60003 &gt; 54.235.188.239:80 [proto: 0/Unknown][1/0 pkts][78/0 bytes]</td>
</tr>
<tr>
<td></td>
<td>Sun Mar 15 20:48:51 2015</td>
<td>Error</td>
<td>TCP SYN Flood</td>
<td>Host 192.168.1.92 is a SYN flooder [77 SYNs sent in the last 3 sec] TCP 192.168.1.92:59751 &gt; 23.223.58.13:80 [proto: 0/Unknown][1/0 pkts][78/0 bytes]</td>
</tr>
<tr>
<td></td>
<td>Sun Mar 15 19:14:38 2015</td>
<td>Error</td>
<td>TCP SYN Flood</td>
<td>Host 127.0.0.1 is under SYN flood attack [255 SYNs received in the last 3 sec] TCP 127.0.0.1:15146 &gt; 127.0.0.1:3000 [proto: 0/Unknown][1/0 pkts][68/0 bytes]</td>
</tr>
<tr>
<td></td>
<td>Sun Mar 15 19:14:38 2015</td>
<td>Error</td>
<td>TCP SYN Flood</td>
<td>Host 127.0.0.1 is a SYN flooder [255 SYNs sent in the last 3 sec] TCP 127.0.0.1:51416 &gt; 127.0.0.1:3000 [proto: 0/Unknown][1/0 pkts][68/0 bytes]</td>
</tr>
<tr>
<td></td>
<td>Sun Mar 15 19:14:38 2015</td>
<td>Error</td>
<td>TCP SYN Flood</td>
<td>Host ::1 is under SYN flood attack [255 SYNs received in the last 3 sec] TCP ::1:51415 &gt; ::1:3000 [proto: 0/Unknown][1/0 pkts][68/0 bytes]</td>
</tr>
<tr>
<td></td>
<td>Sun Mar 15 19:14:38 2015</td>
<td>Error</td>
<td>TCP SYN Flood</td>
<td>Host ::1 is a SYN flooder [255 SYNs sent in the last 3 sec] TCP ::1:51415 &gt; ::1:3000 [proto: 0/Unknown][1/0 pkts][68/0 bytes]</td>
</tr>
<tr>
<td></td>
<td>Sun Mar 15 19:13:27 2015</td>
<td>Error</td>
<td>TCP SYN Flood</td>
<td>Host 127.0.0.1 is under SYN flood attack [115 SYNs received in the last 3 sec] TCP 127.0.0.1:51123 &gt; 127.0.0.1:3000 [proto: 0/Unknown][1/0 pkts][68/0 bytes]</td>
</tr>
</tbody>
</table>
Using ntopng to Trigger Alerts [1/2]

• Wireshark colouring rules cannot be used to trigger alerts and send them to a remote application.

• Instead ntopng can be used as:
  ◦ Data source (e.g. give me the traffic of host X)

• ntopng can use applications (e.g. nagios) for:
  ◦ Sending alerts and state changes.
Using ntopng to Trigger Alerts [2/2]

http://localhost:3000/lua/host_get_json.lua?ifname=eth0&host=192.168.1.1

ntopng (plugin) Active Mode

queueAlert()

ntopng Passive Mode

ntop

nagios

SharkFest 2015 - Computer History Museum June 22-15, 2015
Final Remarks

- ntopng and Wireshark can enable you to implement permanent monitoring while dissecting traffic at packet level.
- Commodity hardware, with adequate software, can now match the performance and flexibility that markets require. With the freedom of open source.
- ntopng and nDPI are available under GNU (L)GPLv3 from https://github.com/ntop