Using nDPI for Monitoring and Security

Luca Deri <deri@ntop.org>
@lucaderi
Who am I

• ntop founder (http://www.ntop.org): company that develops open-source network security and visibility tools:
  ◦ ntopng: web-based traffic monitoring and security
  ◦ nDPI: deep packet inspection toolkit
  ◦ nScrub: software-based DDoS scrubber
  ◦ n2n: peer-to-peer VPN

• Author of various open source software tools.
• Lecturer at the CS Dept, University of Pisa, Italy.
Monitoring Requirements

• Network administrators need to **monitor** and **enforce** network policies hence:
  ◦ Limit the bandwidth of specific protocols (e.g. BitTorrent).
  ◦ Block malicious communications that might travel over encrypted traffic channels.
  ◦ Prioritise specific traffic protocols (e.g. WhatsApp/Skype/Zoom) or cloud protocols.
  ◦ Traffic decryption is not an option: in particular because it is useless, limited to a few protocols (e.g. TLS) and also because it violates the users privacy.
What Do We Want to Accomplish?

• Fingerprint network traffic to detect if both the protocol (e.g. the certificate) has changed or its behaviour.

• Prevent specific traffic flows (e.g. unsafe TLS communications) to happen on our network.

• Provide metrics for measuring the nature of specific communications (e.g. HTTPS) while not being able to inspect the content.

• Identify malware in network communications.
What is Deep Packet Inspection?

• Technique that inspects the packet payload.
• Computationally intensive with respect to simple packet header analysis.
• Concerns about privacy and confidentiality of inspected data.
• Encryption is becoming pervasive, thus challenging DPI techniques.
• No false positives unless statistical methods or IP range/flow analysis are used by DPI tools.
Welcome to nDPI

• In 2012 we decided to develop our own GNU LGPL DPI toolkit (based on a unmaintained project named OpenDPI) in order to build an *opensource* DPI layer.

• Protocols supported exceed 240 and include:
  ◦ P2P (Skype, BitTorrent)
  ◦ Messaging (Viber, Whatsapp, Telegram, Facebook)
  ◦ Multimedia (YouTube, Last.gm, iTunes)
  ◦ Conferencing (Webex, CitrixOnLine)
  ◦ Streaming (Zattoo, Icecast, Shoutcast, Netflix)
  ◦ Business (VNC, RDP, Citrix, Webex)
nDPI in Cybersecurity

nDPI:

- Analyses encrypted traffic to detect issues hidden but un-inspectable payload content.
- Extracts metadata from selected protocols (e.g. DNS, HTTP, TLS..) and matches it against known algorithms for detecting selected threats (e.g. DGA hosts, Domain Generated Algorithm).
- Associates a “risk” with specific flows to identify communications that are affected by security issues.
What is a Protocol in nDPI? [1/2]

• Each protocol is identified as `<major>..<minor>` protocol. Example:
  ◦ DNS.Facebook
  ◦ QUIC.YouTube and QUIC.YouTubeUpload

• Caveat: Skype or Facebook are application protocols in the nDPI world but not for IETF.

• The first question people ask when they have to evaluate a DPI toolkit is: how many protocol do you support? This is not the right question.
What is a Protocol in nDPI? [2/2]

• Today most protocols are HTTP/TLS-based.
• nDPI includes support for string-based protocols detection:
  ◦ DNS query name
  ◦ HTTP Host/Server header fields
  ◦ TLS/QUIC SNI (Server Name Indication)
• Example: NetFlix detection

```cpp
{ "netflix.com", NULL, "netflix" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflxext.com", NULL, "nflxext" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflximg.com", NULL, "nflximg" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflximg.net", NULL, "nflximg" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflxvideo.net", NULL, "nflxvideo" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflxso.net", NULL, "nflxso" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
```
Traffic Classification Lifecycle

• Based on traffic type (e.g. UDP traffic) dissectors are applied sequentially starting with the one that will most likely match the flow (e.g. for TCP/80 the HTTP dissector is tried first).

• Each flow maintains the state for non-matching dissectors in order to skip them in future iterations.

• Analysis lasts until a match is found or after too many attempts (8 packets is the upper-bound in our experience).
nDPI: Packet Processing Performance

nDPI Memory statistics:
- nDPI Memory (once): 203.62 KB
- Flow Memory (per flow): 2.01 KB
- Actual Memory: 95.60 MB
- Peak Memory: 95.60 MB
- Setup Time: 1001 msec
- Packet Processing Time: 813 msec

Traffic statistics:
- Ethernet bytes: 1090890957 (includes ethernet CRC/IFC/trailer)
- Discarded bytes: 247801
- IP packets: 1482145 (of 1483237 packets total)
- IP bytes: 1055319477 (avg pkt size 711 bytes)
- Unique flows: 36703
- TCP Packets: 1338624
- UDP Packets: 143521
- VLAN Packets: 0
- MPLS Packets: 0
- PPPoE Packets: 0
- Fragmented Packets: 1092
- Max Packet size: 1480
- Packet Len < 64: 590730
- Packet Len 64-128: 67824
- Packet Len 128-256: 66380
- Packet Len 256-1024: 157623
- Packet Len 1024-1500: 599588
- Packet Len > 1500: 0
- nDPI throughput: 1.82 M pps / 9.99 Gb/sec
- Analysis begin: 04/Aug/2010 04:15:23
- Analysis end: 04/Aug/2010 18:31:30
- Traffic throughput: 28.85 pps / 165.91 Kb/sec
- Traffic duration: 51367.223 sec
- Guessed flow protos: 0

Single Core (E3 1241v3)
Behaviour and Fingerprinting

• nDPI is not only about application recognition but also:
  ◦ Traffic classification: is this TLS connection a HTTPS connection, a VPN, or something else?
  ◦ Malware recognition: traffic bins (time and packet size)
  ◦ Content enforcement: bytes entropy (measure how bytes are distributed)

<table>
<thead>
<tr>
<th>Server Entropy (SCP)</th>
<th>PDF</th>
<th>PNG</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6,418</td>
<td>7,014</td>
<td>7,008</td>
</tr>
</tbody>
</table>
nDPI: Flow Risks

- HTTP suspicious user-agent
- HTTP numeric IP host contacted
- HTTP suspicious URL
- HTTP suspicious protocol header
- TLS connections not carrying HTTPS (e.g. a VPN over TLS)
- Suspicious DGA domain contacted
- Malformed packet
- SSH/SMB obsolete protocol or application version
- TLS suspicious ESNI usage
- Unsafe Protocol used
- Suspicious DNS traffic
- TLS with no SNI.

- XSS (Cross Site Scripting)
- SQL Injection
- Arbitrary Code Injection/Execution
- Binary/.exe application transfer (e.g. in HTTP)
- Known protocol on non standard port
- TLS self-signed certificate
- TLS obsolete version
- TLS weak cipher
- TLS certificate expired
- TLS certificate mismatch
- DNS suspicious traffic
- HTTP suspicious content
nDPI Encrypted Traffic Analysis

• $ ./example/ndpiReader -J -i ./tests/pcap/instagram.pcap -v 2 -f "port 49355"

TCP 192.168.2.17:49355 --> 31.13.86.52:443 [byte_dist_mean: 125.398474] [byte_dist_std: 67.665465] [entropy: 0.997011] [total_entropy: 5609.185931] [score: 1.0000] [proto: 91.211/TLS.Instagram] [cat: SocialNetwork/6] [456 pkts/33086 bytes --> 910 pkts/1277296 bytes] [Goodput ratio: 9.0/95.3] [14.29 sec] [ALPN: http/1.1] [TLS Supported Versions: TLSv1.3; TLSv1.3 (Fizz)] [bytes ratio: -0.950 (Download)] [IAT c2s/s2c min/avg/max/stddev: 0/0 37.7/0.7 10107/274 546.6/11.8] [Pkt Len c2s/s2c min/avg/max/stddev: 66/66 72.6/1403.6 657/1454 57.2/231.0] [TLSv1.3 (Fizz)] [Client: scontent-mxp1-1.cdninstagram.com] [JA3C: 7a29c223fb122ec64d10f0a159e07996] [JA3S: f4febc55ea12b31ae17cfb7e614afda8] [Cipher: TLS_AES_128_GCM_SHA256]
Detecting Malware

• Clear-text
  ◦ Signatures <- too many signatures, slow.
  ◦ Behaviour <- nDPI (e.g. binary application transfer)

• Encrypted traffic
  ◦ Fingerprint and time/length bins (recognise encrypted traffic patterns)
  ◦ Entropy (speculate about the content nature)
Catching Malware with Fingerprints

• Some malware randomise the clientHello (and thus JA3C) trying to deceive security tools.

• Question: is this a good idea? No, because a monitoring tool will easily detect cases where one IP address features many JA3C fingerprints.
Bytes Entropy [1/2]

• Metric used to measure how bytes are distributed: the larger the entropy, the greater the uncertainty in predicting the value of an observation.

Number of bytes with byte X in the payload

Bytes Entropy [2/2]

• Entropy of raw data before and after encryption (TLS) changes but is it within limited boundaries for homogeneous data.

• Useful to set boundaries on typical protocol entropy and “guess” (up to some extent) the nature of information being exchanged.

### Payload Entropy Distribution

<table>
<thead>
<tr>
<th>Byte Entropy</th>
<th>DNS</th>
<th>TLS</th>
<th>NetFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>4.285</td>
<td>7.789</td>
<td>4.079</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.272</td>
<td>0.231</td>
<td>0.533</td>
</tr>
</tbody>
</table>
Malware Analysis: Trickbot [1/2]

• See https://unit42.paloaltonetworks.com/wireshark-tutorial-examining-trickbot-infections/

• ndpiReader -J -v2 -i 2019-09-25-Trickbot-gtag-ono19-infection-traffic.pcap

• Many TLS flows on non-standard ports, self-signed

TCP 10.9.25.101:49184 <-> 187.58.56.26:449

TCP 10.9.25.101:49165 <-> 144.91.69.195:80
[entropy: 0.000000][total_entropy: 0.000000][score: 0.9943][proto: 7/HTTP][cat: Web/5][203 pkts/11127 bytes <-> 500 pkts/706336 bytes][Goodput ratio: 1/96][5.18 sec][Host: 144.91.69.195][bytes ratio: -0.969 (Download)][IAT c2s/s2c min/avg/max/stddev: 0/0 319/365 49/37][Pkt Len c2s/s2c min/avg/max/stddev: 54/54 55/1413 207/1514 11/134][URL: 144.91.69.195/solar.php][StatusCode: 200][ContentType: application/octet-stream][UserAgent: pwtyyEKzNtGatwnJjmCcBLbOveCVpC][Risk: ** Binary application transfer **][PLAIN TEXT (GET /solar.php HTTP/1.1)]
Malware Analysis: Trickbot [2/2]

• Same packet sequence, same packet len and time distribution (using bins to detect similarities), same entropy…
https://github.com/ntop/nDPI