Demo: Simple Deep Packet Inspection with P4

Sahil Gupta, Devashish Gosain,Garegin Grigoryan
Minseok Kwon, H. B. Acharya

Background

- NIDS and Firewalls
  - Analyse network traffic to detect malicious activity
  - Monitor system calls, application layer protocol events, inspect packet content, etc.
  - Reconnaissance, Access control, DDoS, Data leakage, Malware detection

- Deep Packet Inspection
  - Lawful interception, Copyright enforcement, Surveillance / Filtering, etc.
  - Also used: Statistics, Quality of service.

- Pattern matching cases in DPI
  - Basic String matching (DFA without loops)
  - Regular expression (NFA/DFA)
  - PCRE

Previous works

- Hula [SOSR 2016],
  Poseidon [NDSS 2020],
  Gallium [Sigcomm 2021],
  F. Paolucci [JOCN 2019]
  - Port scan attack detector, DDoS attack detector
  - Firewall, NAT Network function, Load balancer,
  - Proxy, Flow based trojan Detector

- None of these make use of DPI functionality.
  - In fact, $P4_{16}$ standard makes it clear, P4 is not meant to do DPI.

Application Layer Firewalling?

- Simple DPI : string matching.
  - This is enough to detect domain name.
    - TLS client hello (SNI field), DNS request (qname field), HTTP GET request (Host field)
  - Use case: ISP and governments, to filter unethical websites.
    - Child pornography, illegal guns/drugs trading websites, etc.

- Motivations.
  - Testing feasibility of DPI in the dataplane.
  - Possible: saving bandwidth and compute power
    - No east-west traffic (sending packet to control plane) for DPI.
  - Possible: improved accuracy
    - All packets inspected in dataplane itself. Not a sample.

Challenges

- Protocols headers (TLS, DNS, HTTP) are highly flexible
    - optional fields
    - variable field ordering
    - variable-length fields

- Data required
    - start and end location of the domain name (as offset in the packet).

Research Question.

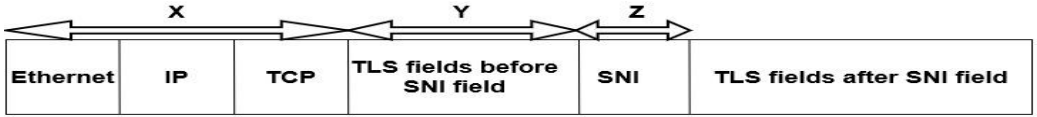
In spite of these above challenges,

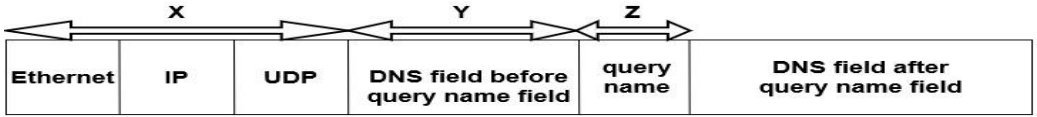
Is it possible to perform (simple) deep-packet inspection in the data plane and reliably detect URLs in traffic,

in all practically significant cases, and for multiple important protocols,

using only standard P4-compliant switches?

Approach





Browser vs OS	Mozilla firefox	Google Chrome	Microsoft Edge
Windows 10	Total packets: 1743 Offset Y = 127 bytes	Total packets: 3216 Offset Y = 127 bytes	Total packets: 2812 Offset Y = 127 bytes
Linux Ubuntu 18.04 LTS	Total packets: 3230 Offset Y = 127 bytes	Total packets: 1307 Offset Y = 127 bytes	Total packets: 1432 Offset Y = 127 bytes

Browser vs OS	Mozilla firefox	Google Chrome	Microsoft Edge
Windows 10	Total packets: 512 Offset Y = 13 bytes	Total packets: 5587 Offset Y = 13 bytes	Total packets: 4237 Offset Y = 13 bytes
Linux Ubuntu 18.04 LTS	Total packets: 42238 Offset Y = 13 bytes	Total packets: 5075 Offset Y = 13 bytes	Total packets: 6777 Offset Y = 13 bytes

Experimental Setup (details in video)

- Network topology (emulated using Mininet).
  - P4 switch S1 (emulated using standard BMV2 model) connects two hosts H1 and H2.
  - H1 generates mixed censored and benign traffic to H2
  - H2 runs HTTPS, DNS, and HTTP services
- Experiment targets
  - TCAM MA rules are installed by the control plane.
  - We measure how varying the number of filtered patterns will affect:
    - Switch Delay
    - Web response time
- Additional python scripts
  - To compute results from mininet generated pcap files
  - To automatically generate TCAM Match-Action-Table rules to filter connections.

Concluding Remarks.

- Simple basic DPI can be performed by standard (P4-compliant) switches
- Future work
  - Demonstrate simple DPI on a real P4 switch
  - Large scale study to provide starting and ending of domain name in each kind of packet with high accuracy (Alexa top 10,000 websites)
  - Developing a more advanced approach to inspect entire TCP/UDP payload
- Demo