CLayer

Packet classification with explicit coordination

Mosharaf Chowdhury with Sameer Agarwal, Dilip Joseph, and Ion Stoica
Motivation

Packet classification is everywhere

- Link (2.5)
  - Switching, MPLS
- Network
  - Forwarding
- Transport
  - Filtering, IntServ, DiffServ
- Application
  - Load balancing, Intrusion detection

January 22, 2010  Google GRAD CS Forum @ Mountainview
Problems

Existing approaches are point solutions for specific layer/service

Packet classification is expensive
  » Computation and memory
  » Power hungry

Configuration complexity
  » Lack of coordination between entities involved

Semantic gap
Solution

CLayer is a cross-layer classification primitive

» Generic mechanism to configure and implement capability-driven classification offloading
» Explicit coordination between classifiers and helpers

“Classify once, verify thereafter”

Label-based per-flow classification

» Labels are verifiable, confidential, and non-transferable
Outline

CLayer classification model
Fate-carrying labels (FCLs)
Implementation
Results
Classification model
Control plane

End host A -> QoS enabled router -> Load balancer

2a. ClassReq: QoS:q1
2b. ClassReq: QoS:q1, WebSess:1
3. ClassRsp: QoS:q1, WebSess:1

Web server 1 -> Load balancer
Web server 2
Data plane

End host A → QoS enabled router → Load balancer → Web server 1, Web server 2

- QoS: q1
- WebSess: 1
- Payload

January 22, 2010
Fate-carrying labels
FCL basics

A *label* in CLayer is an opaque bag of bits
  » Issued by a classifier for a particular flow
  » Meaningful only to the issuer
  » `<label → action>` lookup

A fate-carrying label carries the action itself
  » No `<label → action>` lookup
  » No states in classifiers
Requirements

Authenticity and Integrity
» Verifiable and non-transferable
» Unforgeable and single-use only

Confidentiality
» Impossible to infer

Performance
» Not better off without CLayer

HMAC
Checksum

Obfuscation
Periodic Invalidation

Line-speed hashing
Low overhead
Placement

Application Layer
Transport Layer
CLayer
Network Layer

**CLayer Header**

<table>
<thead>
<tr>
<th>N</th>
<th>MSG</th>
<th>HANDLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INFO (N – 1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TYPE</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEN</td>
<td>CHECKSUM</td>
</tr>
<tr>
<td></td>
<td>ACTION</td>
</tr>
</tbody>
</table>

HMAC (5-tuple, ACTION, SECRET)

FCL

January 22, 2010
Implementation
Implementation stats

C++ Implementation using user level Click software router

Core components:
  » CLayer socket library and daemon (4025 lines)
  » Layer 4 firewall (308 lines)
  » Layer 4 load balancer (190 lines)

Ported applications:
  » lighttpd, httpperf, wget, nuttcp, elinks (< 50 lines)
Results
Overheads

CLayer overheads at *helpers*:
- State: \(~10\) bytes per connection
- Processing: less than 1 \(\mu\)s

At *classifiers*:
- No state overheads
- Processing: varies in s/w and h/w implementations

Per-packet overheads:
- Proportional to the number of labels
- Potential bottleneck
Performance

![Graph showing performance over number of rules with a red line for Regular-Snort and a blue dotted line for CLayer-Snort. The graph highlights an attractiveness threshold at around 15 MB/s.](image)
Multiple classifiers

![Graph showing goodput (MB/s) vs. number of firewalls. Each firewall has 7500 rules. The graph compares CLayer with and without HMAC, highlighting hash and software implementation overhead.]

January 22, 2010

Google GRAD CS Forum @ Mountainview
Summary

Packet classification requires a dedicated layer

CLayer provides significant performance gain
  » 2-4 times increase in classifier throughput
  » Additional ~100% increase in throughput in trusted domains or with line-speed h/w hashing

CLayer adoption requires minimal change
  » Most suitable for controlled environments like data center and enterprise networks
Questions
Backup
CLayer handshaking

1. End host A
   - CL_SYN
     - Capability: [A, TCPFlow, Label]

2. Router E
   - CL_SYN
     - Capability: [A, TCPFlow, Label]

3. End host B
   - CL_SYN
     - Capability: [A, TCPFlow, Label]

4. End host A
   - CL_SYNACK
     - Capability: [B, TCPFlow, Label]
     - EchoReq: [E, A, TCPFlow, Label:q]
     - ClassReq: [E, B, TCPFlow, Label:q]

5. Router E
   - CL_SYNACK
     - Capability: [B, TCPFlow, Label]

6. End host B
   - CL_SYNACK
     - Capability: [B, TCPFlow, Label]
     - EchoReq: [E, A, TCPFlow, Label:q]
     - ClassReq: [E, B, TCPFlow, Label:q]

7. End host A
   - CL_ACK2
     - Results: [E, TCPFlow, Label:q]

8. Router E
   - CL_ACK2
     - Results: [E, TCPFlow, Label:q]

Results:
- [E, TCPFlow, Label:q]