**P4 Code Implementation for Feature Extraction**

|  |
| --- |
| **// P4 program for Data-Plane-enabled Feature Extraction**  // P4 Header Definitions  header ethernet {  address srcAddr;  address dstAddr;  short etherType;  }  header ip {  bit protocol;  address srcAddr;  address dstAddr;  }  header tcp {  short srcPort;  short dstPort;  ... other TCP header fields ...  }  header icmp {  bit type;  ... other ICMP header fields ...  }  // P4 Table Definitions  table flow\_table {  fields {  ingress\_port: ingress\_port\_t;  ipv4\_srcAddr: ipv4\_address;  ipv4\_dstAddr: ipv4\_address;  tcp\_srcPort: tcp\_port;  tcp\_dstPort: tcp\_port;  in\_metadata: metadata; // Stores packet arrival timestamp  }  actions {  add\_to\_feature\_table;  }  }  table feature\_table {  fields {  flow\_id: handle\_t; // Foreign key referencing flow\_table entry  wpc: counter;  bytesw: counter;  pps: varbit;  bps: varbit;  df: bool;  ... other features from Table 2 ...  }  }  table attack\_state\_table {  fields {  flow\_id: handle\_t; // Foreign key referencing flow\_table entry  state: enum { NORMAL, POTENTIAL\_ATTACK, CONFIRMED\_ATTACK };  }  // P4-based Feature Extraction  register<counter> packet\_counter;  table\_match my\_table\_match {  type: exact;  fields {  standard\_metadata.ingress\_port: ingress\_port\_t;  ethernet.etherType: short;  }  }  table\_action my\_table\_action {  actions {  packet\_counter.increment();  flow\_table.add\_to\_feature\_table;  }  }  struct feature\_data {  handle\_t flow\_id;  ... other features from Table 2 ...  }  packet ingress(packet in\_packet)  // Extract header information  var ethernet eth;  var { ip, tcp } hdr;  parse(in\_packet, eth, hdr);  // Match on ingress port and etherType  if (my\_table\_match(standard\_metadata.ingress\_port, eth.etherType)) {  packet\_counter.increment();  // Flow lookup (assuming source and destination IP/port for simplicity)  table\_match flow\_lookup\_match {  type: exact;  fields {  standard\_metadata.ingress\_port: ingress\_port\_t;  ip.srcAddr: ipv4\_address;  ip.dstAddr: ipv4\_address;  tcp.srcPort: tcp\_port;  tcp.dstPort: tcp\_port;  }  }  action flow\_lookup\_action {  flow\_table.add\_to\_table(  ingress\_port: standard\_metadata.ingress\_port,  ipv4\_srcAddr: ip.srcAddr,  ipv4\_dstAddr: ip.dstAddr,  tcp\_srcPort: tcp.srcPort,  tcp\_dstPort: tcp.dstPort,  in\_metadata: ingress\_metadata // Store timestamp  );  }  apply(flow\_lookup\_match, flow\_lookup\_action);  // Send features to cloud server for analysis by the ensemble model  send\_to\_controller(features);  } |