Here's a complete P4 program for deploying the P4-Enabled Multi-Control Adaptive Mitigation (P4-MCAM) algorithm. This implementation represents the mitigation module, which is deployed at the P4 switch.

**Implementation of the P4-MCAM Algorithm as a P4 Application**

|  |
| --- |
| **// Define the headers**  **header ethernet\_t {**  **macAddr\_t dstAddr;**  **macAddr\_t srcAddr;**  **bit<16> ethType;**  **}**  **header ipv4\_t {**  **bit<4> version;**  **bit<4> ihl;**  **bit<8> diffserv;**  **bit<16> totalLen;**  **bit<16> identification;**  **bit<3> flags;**  **bit<13> fragOffset;**  **bit<8> ttl;**  **bit<8> protocol;**  **bit<16> hdrChecksum;**  **ip4Addr\_t srcAddr;**  **ip4Addr\_t dstAddr;**  **}**  **header tcp\_t {**  **bit<16> srcPort;**  **bit<16> dstPort;**  **bit<32> seqNo;**  **bit<32> ackNo;**  **bit<4> dataOffset;**  **bit<6> reserved;**  **bit<6> flags;**  **bit<16> window;**  **bit<16> checksum;**  **bit<16> urgentPtr;**  **}**  **// Define the metadata**  **struct metadata\_t {**  **bit<1> is\_attack;**  **}**  **// Define the parser**  **parser MyParser(packet\_in packet,**  **out ethernet\_t eth\_hdr,**  **out ipv4\_t ip\_hdr,**  **out tcp\_t tcp\_hdr,**  **inout metadata\_t meta) {**  **state start {**  **packet.extract(eth\_hdr);**  **transition select(eth\_hdr.ethType) {**  **0x0800: parse\_ipv4;**  **default: accept;**  **}**  **}**  **state parse\_ipv4 {**  **packet.extract(ip\_hdr);**  **transition select(ip\_hdr.protocol) {**  **6: parse\_tcp;**  **default: accept;**  **}**  **}**  **state parse\_tcp {**  **packet.extract(tcp\_hdr);**  **transition accept;**  **}**  **}**  **// Define tables**  **table attack\_detection {**  **key = {**  **ip\_hdr.srcAddr: exact;**  **ip\_hdr.dstAddr: exact;**  **tcp\_hdr.srcPort: exact;**  **tcp\_hdr.dstPort: exact;**  **}**  **actions = {**  **set\_attack\_flag;**  **no\_op;**  **}**  **size = 1024;**  **default\_action = no\_op();**  **}**  **table mitigation\_actions {**  **key = {**  **meta.is\_attack: exact;**  **}**  **actions = {**  **drop\_packet;**  **forward\_packet;**  **}**  **size = 2;**  **default\_action = forward\_packet();**  **}**  **// Define actions**  **action set\_attack\_flag() {**  **meta.is\_attack = 1;**  **}**  **action no\_op() {**  **// Do nothing**  **}**  **action drop\_packet() {**  **mark\_to\_drop();**  **}**  **action forward\_packet() {**  **// Forward packet to the next hop (default action)**  **}**  **// Define the control block**  **control MyIngress(inout ethernet\_t eth\_hdr,**  **inout ipv4\_t ip\_hdr,**  **inout tcp\_t tcp\_hdr,**  **inout metadata\_t meta) {**  **apply {**  **attack\_detection.apply();**  **mitigation\_actions.apply();**  **}**  **}**  **// Define the deparser**  **control MyDeparser(packet\_out packet,**  **in ethernet\_t eth\_hdr,**  **in ipv4\_t ip\_hdr,**  **in tcp\_t tcp\_hdr) {**  **apply {**  **packet.emit(eth\_hdr);**  **packet.emit(ip\_hdr);**  **packet.emit(tcp\_hdr);**  **}**  **}**  **// Define the pipeline**  **control MyPipeline {**  **MyParser() parser;**  **MyIngress() ingress;**  **MyDeparser() deparser;**  **apply {**  **parser.apply();**  **ingress.apply();**  **deparser.apply();**  **}**  **}**  **MyPipeline() main;** |

**Explanation of the P4 Program:**

1. **Headers Definition**: The program starts by defining the Ethernet, IPv4, and TCP headers, which will be extracted from the incoming packets.
2. **Metadata Definition**: A metadata structure ***metadata\_t*** is defined, which contains a single bit ***is\_attack*** to flag whether a packet is considered part of an attack.
3. **Parser**: The ***MyParser*** function extracts the Ethernet, IPv4, and TCP headers from incoming packets. It starts by extracting the Ethernet header and then transitions based on the ***ethType*** to either extract IPv4 and TCP headers or accept the packet.
4. **Tables**:

* **attack\_detection**: This table uses the source IP, destination IP, source port, and destination port as keys to identify known attack patterns. If an attack is detected, the ***set\_attack\_flag*** action is applied.
* **mitigation\_actions**: This table checks the ***is\_attack*** flag in the metadata. If the flag is set, the packet is dropped; otherwise, it is forwarded as usual.

1. **Actions**:

* **set\_attack\_flag**: Sets the ***is\_attack*** metadata field to 1, indicating that the packet is part of an attack.
* **no\_op**: Does nothing, used as a default action.
* **drop\_packet**: Marks the packet to be dropped.
* **forward\_packet**: Forwards the packet to the next hop, the default action for non-attack packets.

1. **Control Block (MyIngress)**: This control block applies the ***attack\_detection*** and ***mitigation\_actions*** tables. Based on the outcomes, the packet is either forwarded or dropped.
2. **Deparser**: The ***MyDeparser*** function reassembles the packet from the Ethernet, IPv4, and TCP headers before sending it out.
3. **Pipeline**: The ***MyPipeline*** control connects the parser, ingress, and ***deparser***, forming the packet processing pipeline.

**Step-by-Step Deployment Instructions:**

1. **Setup P4 Development Environment**: Ensure that we have the P4 development environment set up, including tools like P4C (P4 compiler) and BMv2 (Behavioral Model v2).
2. **Write the P4 Program**: Copy the P4 code provided above into a ***.p4*** file (***e.g., p4\_mcam.p4***).
3. **Compile the P4 Program**: Use the P4C compiler to compile the P4 program: ***p4c --target bmv2 --arch v1model p4\_mcam.p4 -o p4\_mcam.json***
4. **Deploy on a P4 Switch**: Load the compiled JSON file onto your P4-enabled switch or a BMv2 software switch: ***simple\_switch --log-console --json p4\_mcam.json***
5. **Configure the Switch**: Use a control plane application or a simple runtime CLI to populate the ***attack\_detection*** and ***mitigation\_actions*** tables with appropriate entries.
6. **Monitor and Adapt**: Continuously monitor the network using the deployed P4-MCAM module. Adjust the control plane logic and table entries as new attack patterns are identified or as the network environment changes.
7. **Logging and Reporting**: Ensure that logs are generated for all mitigation actions, including packet drops and forwarding decisions, and store them for future analysis.