**ACM Module Source Code**

|  |
| --- |
| **// Header Definitions**  **header\_type ethernet\_t {**  **fields {**  **dst\_addr : 48;**  **src\_addr : 48;**  **eth\_type : 16;**  **}**  **}**  **header\_type ipv4\_t {**  **fields {**  **version : 4;**  **ihl : 4;**  **diffserv : 8;**  **totalLen : 16;**  **id : 16;**  **flags : 3;**  **fragOffset : 13;**  **ttl : 8;**  **protocol : 8;**  **hdrChecksum: 16;**  **srcAddr : 32;**  **dstAddr : 32;**  **}**  **}**  **// Metadata and Control Registers**  **register PTPW : 32; // Packet Throughput within Time Window**  **register SAPR : 32; // Suspicious Activity Packet Rate**  **register FER : 32; // Flow Error Rate**  **register FPS : 32; // Frequency of Packets Sent**  **// Tables for Detection and Mitigation**  **table detect\_attack {**  **reads {**  **PTPW : exact;**  **SAPR : exact;**  **FER : exact;**  **FPS : exact;**  **}**  **actions { detect\_anomaly; no\_op; }**  **}**  **table mitigate\_attack {**  **reads {**  **ipv4.srcAddr : exact;**  **ipv4.dstAddr : exact;**  **}**  **actions { drop; rate\_limit; }**  **}**  **// Actions for Attack Detection and Mitigation**  **action detect\_anomaly() {**  **// Logic to identify attack patterns using thresholds**  **if (PTPW > threshold\_ptpw) {**  **mark\_as\_suspicious();**  **}**  **if (SAPR > threshold\_sapr) {**  **mark\_as\_suspicious();**  **}**  **if (FER > threshold\_fer) {**  **mark\_as\_suspicious();**  **}**  **if (FPS > threshold\_fps) {**  **mark\_as\_suspicious();**  **}**  **}**  **action mark\_as\_suspicious() {**  **metadata.suspicious\_flag = 1;**  **}**  **action drop() {**  **drop();**  **}**  **action rate\_limit() {**  **apply\_rate\_limit();**  **}**  **action no\_op() {}**  **// Main Control Logic**  **control ingress {**  **apply(detect\_attack);**  **if (metadata.suspicious\_flag == 1) {**  **apply(mitigate\_attack);**  **}**  **}** |

**Explanation of the P4 Code:**

1. **Header Definitions**: Defines Ethernet and IPv4 headers, which are the fundamental network layers this module will process.
2. **Metadata and Control Registers**: Registers (PTPW, SAPR, FER, FPS) represent traffic metrics crucial for detecting anomalies. These values are computed at the control plane and updated on the switch, providing data for assessing suspicious activities.
3. **Detection Table (detect\_attack)**: Examines traffic metrics against predefined thresholds, checking if any of the PTPW, SAPR, FER, or FPS metrics exceed safe values. If so, the *detect\_anomaly* action is applied, marking the packet as suspicious.
4. **Mitigation Table (mitigate\_attack)**: Determines mitigation actions for traffic marked as suspicious. The drop action discards packets from flagged sources, while *rate\_limit* reduces bandwidth for potentially malicious sources.
5. **Actions**: The *detect\_anomaly* action checks traffic metrics against thresholds, marking suspicious flows for closer inspection. The drop and *rate\_limit* actions then mitigate threats by filtering or limiting suspicious traffic.
6. **Control Logic**: The ingress control block first applies *detect\_attack* to assess the flow's safety. If a flow is suspicious, *mitigate\_attack* applies mitigation based on predefined rules, isolating potentially harmful traffic.

**Step-by-Step Deployment Instructions (Emulation in Mininet-WiFi):**

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., ACM.p4***).
3. **Compile the P4 Program**: Use the P4C compiler to compile the P4 program: ***p4c --target bmv2 --arch v1model ACM.p4 -o ACM.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 ACM.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 ACM module. Adjust the control plane logic and table entries as new attack patterns are identified or as the network environment changes.
7. **Observe ACM Responses**: Monitor packet drop and rate-limiting responses on the P4 switch in real-time by viewing packet stats.
8. **Logging and Reporting**: Ensure that logs are generated for all mitigation actions, including packet drops and forwarding decisions, and store them for future analysis.

**Step-by-Step Deployment Instructions (Real-World Deployment):**

1. **Select a P4-Enabled Switch**: Choose a P4-compatible switch (e.g., Intel Tofino) for real-world deployment. Ensure the hardware meets the requirements for implementing real-time traffic adaptation.
2. **Install P4 Runtime Environment**: Install the P4 software environment on the network (e.g., Barefoot’s Tofino SDK for Intel switches).
3. **Compile and Load P4 Program**: Compile the ***acm.p4*** file using the switch’s SDK. Load the compiled program onto the P4 switch.
4. **Integrate with SDN Controller**: Connect the P4 switch to an SDN controller (e.g., ONOS or Floodlight). Ensure the control plane has policies to adjust thresholds dynamically and respond to alerts from the ACM module.
5. **Configure Secure Communication Channels**: Establish secure, encrypted channels for controller-switch communication to coordinate real-time alerts and responses.
6. **Run and Test Traffic Scenarios**: Use production-grade traffic monitoring tools (e.g., *Wireshark*) and simulate attack traffic to observe real-time packet handling, mitigation, and coordination across data and control planes.
7. **Dynamic Mitigation:** This setup ensures the ACM module can dynamically adjust mitigation levels, providing responsive and cooperative defense against evolving network threats.