ASID Framework Deployment Manual

Version: 1.0 (November 2025)

1. Overview

The ASID Framework is a programmable, multi-controller security architecture for Software-Defined Networks (SDN). It integrates in switch P4 feature extraction, machine-learning-based detection, and distributed mitigation across RYU, ONOS, and OpenDaylight controllers.

2. System Requirements

• Operating System: Ubuntu 20.04 or 22.04 (64-bit)
• Python: Version ≥ 3.9
• SDN Controllers:
  - RYU v4.34+
  - ONOS 2.7+
  - OpenDaylight Aluminium or later
• Mininet 2.3+
• BMv2 (Behavioral Model v2) with P4Runtime and gRPC support
• P4C compiler
• Prometheus & Grafana (for monitoring)
• Git, pip, and basic networking tools (curl, netcat, etc.)

3. Directory Structure

```
ASID/
├── controllers/
├── dataplane/
├── detection/
├── mitigation/
├── experiments/
└── utils/
```

4. Environment Setup

1. Install dependencies:
   sudo apt update && sudo apt install git python3-pip mininet p4lang-p4c -y

2. Clone the ASID repository:
   git clone https://github.com/<yourusername>/ASID_Framework.git
   cd ASID_Framework

3. Install Python libraries:

pip install -r requirements.txt

4. Ensure BMv2 and Mininet are correctly configured and accessible from PATH.

5. Deploying the P4 Dataplane

1. Compile P4 program:
p4c --target bmv2 --arch v1model -o dataplane/runtime
dataplane/p4src/asid_traffic_engineering.p4

2. Launch BMv2 switch with gRPC:
simple_switch_grpc --device-id 1 --no-p4 --p4info
dataplane/runtime/p4runtime_shell_config.json --config
dataplane/runtime/asid_pipeline.bmv2.json

3. Verify switch operation via Mininet or BMv2 CLI.

6. Controller Configuration

RYU:
ryu-manager controllers/ryu/ryu_controller.py &

ONOS:
onos-service start
python3 controllers/onos/onos_app.py &

OpenDaylight:
./distribution-karaf/bin/karaf
python3 controllers/opendaylight/odl_app.py &

7. Running Detection and Mitigation

1. Start the CAAWE ensemble detection engine:
python3 detection/caawe_ensemble.py &

2. Start the Distributed Multi-Controller Mitigation Core (DMCM):
python3 mitigation/dmcm_core.py &

3. Logs are stored in mitigation/logs/mitigation_log.json.

8. Mininet Topology and Testing

1. Launch the ASID Mininet topology:
   sudo python3 experiments/mininet_topology.py

2. Generate test traffic:
   python3 experiments/traffic_generator.py

3. Evaluate system performance:
   python3 experiments/performance_evaluator.py

9. Monitoring and Visualization

1. Configure Prometheus (experiments/grafana_prometheus_config/prometheus.yml)
2. Start Prometheus and Grafana:
   docker-compose -f experiments/grafana_prometheus_config/docker-compose.yml up
3. Open Grafana in browser (http://localhost:3000) and import grafana_dashboard.json.

10. Model Retraining and Maintenance

• Retrain detection models on new data:
  python3 detection/retraining/feedback_loop.py

• Incrementally update ensemble weights:
  python3 detection/retraining/incremental_update.py

• Regularly clean logs and export summaries from experiments/ folder.

11. Troubleshooting

Common Issues:
- Ensure ports 5005–5007 are free before launching controllers.
- Verify ONOS and ODL REST APIs are reachable.
- If detection accuracy drops, clear PASD.csv and re-extract features.
- For Mininet errors, restart the network with 'sudo mn -c'.