Policy monitoring and enforcement are an important aspect of network security. They enable administrators to prevent hackers and malware from compromising a network and stealing data.

However, today’s methods of defining network policies manually are too slow to adapt to configuration changes in cloud networks. Manually configuring networks to enforce policy is a time-consuming, error-prone process. Systems for enforcing policy in cloud networks must react to changing topologies and conditions faster than a human user can.

**Goals**

- Enforce network-level dynamic policies
- Allow administrators to define high-level dynamic policies that apply to classes of hosts
- Use logical inference to let the system deduce runtime actions from abstract policy
- Permit the construction of policies that can alter network traffic flow in response to network changes and threats
- React and adapt to changes in cloud networks more quickly than a human

**Fundamental Questions/Challenges**

- Network traffic flow is controlled by many different components spread across the network
- Programming these components to enforce network-level policy manually is slow and error-prone
- Cloud networks are highly dynamic: new hosts are constantly joining the network
- Hosts in cloud networks may have dynamic security requirements
- Host-level monitoring systems can’t detect network policy violations

**Research Plan**

- Research existing network-level policy enforcement systems as related work
- Research OpenFlow and NOX for controlling network traffic flow
- Research Jena inference engine for applying policies
- Build NetOdessa system: a network-level dynamic policy enforcement system
- Test NetOdessa under typical network loads

**Research Results**

NetOdessa components:

- **Flow Sentinel** Uses NOX to monitor and control connections
- **Verifier** Uses Jena’s inference engine to deduce actions from policies and apply actions to flows

**Experimental setup:**

- Experiment 1: Enforce 1 rule, varying the volume of network traffic
- Experiment 2: Fix the network traffic volume, varying the number of rules enforced

**Related Work/Interaction with Other Projects**

NetOdessa is built on several technologies:

- **Odessa**: a distributed host-level policy-monitoring system. NetOdessa is inspired by and designed after Odessa.
- **OpenFlow**: an open protocol for controlling commodity switches.
- **NOX**: a “network operating system” that gives applications an interface for controlling OpenFlow-compatible switches.
- **RDF (Resource Description Framework)**: a general description language.
- **Jena**: a Java library for reasoning about systems described in RDF.

Related work includes:

- **FSL/FML (Flow-based Security/Management Language)**: a centralized network management system that enforces static, high-level network policies.
- **Resonance**: a state-based system for enforcing access control in networks based on the **limited BLP model**.