CNaaS NAC - Installation

Introduction

CNaaS NAC provides a way for clients to authenticate themselves using IEEE 802.1X and/or MAB.

Features:

- Automatic registration of MAB clients.
- Periodic cleanup of inactive clients.
- Replication between primary and secondary server.
- API which can be used for all sorts of integrations.
- Port-locking, possible to bind clients to a single switch port for enhanced security.
- Fancy web UI written in React.

CNaaS NAC consists of two parts, NAC and NAC frontend, both runs in Docker.

Docker

We are using Docker Compose to manage all containers. First of all we want to start the CNaaS NAC containers without the frontend and a minimal YAML-file can look something like this (of course secrets and passwords should be replaced):

```
version: '3.7'
services:
  nac_api:
    build: ./api/
    ports:
      - 1443:4430
    networks:
      cnaas:
        ipv4_address: 172.31.0.10
    depends_on:
      - nac_postgres

  nac_radius:
    build: ./radius
    ports:
      - 1812:1812/udp
      - 1813:1813/udp
    networks:
      cnaas:
        ipv4_address: 172.31.0.20
    environment:
      - RADIUS_SERVER_SECRET=testing123
      - RADIUS_NO_PORT_LOCK=True
    depends_on:
      - nac_api
    volumes:
      - type: volume
        source: nac-freeradius-data
        target: /etc/Freeradius/3.0/
      - type: volume
        source: nac-var-data
        target: /var/
      - type: volume
        source: nac-api-certs
        target: /opt/cnaas/certs/

  nac_postgres:
    build: ./postgres
    volumes:
      - type: volume
        source: nac-postgres-data
        target: /var/lib/postgresql/data/
    environment:
      - POSTGRES_USER=cnaas
      - POSTGRES_PASSWORD=cnaas
      - POSTGRES_DB=nac
    ports:
      - 5432:5432
```

```
    networks:
      cnaas:
        ipv4_address: 172.31.0.30

networks:
  cnaas:
    driver: bridge
    name: cnaas
    ipam:
      config:
      - subnet: 172.31.0.0/24
    driver_opts:
      com.docker.network.bridge.name: br-cnaas

volumes:
  nac-postgres-data:
    external: false
  nac-freeradius-data:
    external: false
  nac-var-data:
    external: false
  nac-api-certs:
    external: false
```

This will launch the containers needed for NAC. If you need to edit any configuration files for FreeRADIUS (out of the scope of this document) the easiest is to either run bash inside the nac_radius container and edit the files or do it from the host and use the directory in which nac-freeradius-data is mounted. Control socket etc for radmin is enabled inside the container for debugging.

To launch the frontend we must first have CNaaS Auth POC running, instructions available here: CNaaS Auth POC server installation. The compose file f or CNaaS NAC Front looks like this:

```
version: '3.7'
services:
  cnaas_front:
    build: ./front/
    ports:
      - 4443:4443
    environment:
      - NAC_API_URL=https://url-to-nac-api/
      - NAC_FRONT_URL=https://url-to-cnaas-front/
      - AUTH_API_URL=https://url-to-cnaas-auth-poc/
    volumes:
      - type: volume
        source: cnaas-front-cert
        target: /opt/cnaas/cert
volumes:
  cnaas-front-cert:
    external: true
```

The three variables above should point to the URL the NAC API container exposes, the URL to the CNaaS NAC Front container and finally URL to the auth server. This works in the same way as for CNaaS NMS.