netengine Documentation

Release 0.1

Alessandro Bucciarelli, Federico Capoano

Jun 13, 2021

Contents

Netengine is a python library that aims to provide a single API to extract common information from network devices with different firwmares (eg: OpenWRT, AirOS) using different protocols such as the Simple Network Management Protocol (SNMP), and the ability to easily add other backends like SSH and HTTP (*read more*).

You can immagine **Netengine** as a read-only ORM (Object Relational Mapper) equivalent for networks.

CHAPTER 1

Motivations

While dealing with networks in the real world, it's highly probable that you will deal with a network which is made with very different routers, switches and servers. Some may support standard SNMP mibs, some may not, some may implement other HTTP APIs, some may even implement obscure/custom SNMP mibs.

If you need to develop a web application that automates some networking tasks, you don't want to deal with all those differences in the application code, because it would become hard to mantain very soon. You also might not want to tie your web app code to a specific vendor or firmware because that would make your software unflexible.

If we had a single API we could let web developers focus on the task they need to accomplish rather than dealing with different firmwares, different linux distributions and so on.

The goal of this project is to build that single API.

CHAPTER 2

Status of this project

We are currently in 0.1 alpha version.

The 0.1 final version will be out by August 2021.

Note: The legacy versions of this project had support for SSH and HTTP for extracting information from devices. To see how it worked, visit the 0.1.0 alpha release page on github.

CHAPTER 3

Install

Install the development version (tarball):

```
pip install https://github.com/openwisp/netengine/tarball/master
```

Alternatively, you can install via pip using git:

```
pip install -e git+git://github.com/openwisp/netengine#egg=netengine
```

CHAPTER 4

Contents:

4.1 Usage

The usage of Netengine module requires it to be installed properly as explained in *index*. If you have an installation under a virtualenv, enter the folder /bin and type:

```
source activate
```

otherwise (if you have installed globally) just open an editor as bpython and you we are ready to go.

These are the main steps to follow to use the module:

- import the correct backend and supported framework
- declare a device using the proper constructor
- invoke methods over the device just declared

So we have:

```
from netengine.backends.<backend_name> import <supported_firmware>

<device_name> = supported_firmware_constructor
```

To invoke methods over the just declared device it's necessary to use the dot notation as:

```
<device_name>.<method or property>
```

Further example will be found inside dedicated docs for every backend

4.2 Running tests

Install test reqirements:

```
pip install -r reqirements.txt
pip install -r requirements-test.txt
```

Clone repo:

```
git clone git://github.com/openwisp/netengine

./runtests.py
```

To run tests on real devices, first copy the settings file:

```
cp test-settings.example.json test-settings.json
```

Then change the credentials accordingly, now run tests with:

```
DISABLE_MOCKS=1 TEST_SETTINGS_FILE='test-settings.json' ./runtests.py
```

See test coverage with:

```
nose2 --with-coverage
```

Run specific tests by specifying the relative path:

```
# base tests
nose2 tests.base

# snmp tests
nose2 tests.snmp
# snmp openwrt specific tests
nose2 tests.snmp.openwrt

# run without mocks with a custom test file
DISABLE_MOCKS=1 TEST_SETTINGS_FILE='test-settings.json' nose2 tests.snmp
```

4.3 SNMP backend

4.3.1 SNMP

SNMP (Simple Network Management Protocol) is a network protocol very useful for retrieving info from a device. All the information is retrieved by using codes called MIBs. All MIBs have a tree-like structure, every main information is the root and by adding more detail to the info the tree gains more depth. Obviously, by getting the smallest MIB which is “1” or simply ” . ” one can get all the tree.

The SNMP backend provides support for 2 firmwares:

- AirOS
- OpenWRT

4.3.2 AirOS example

```
from netengine.backends.snmp import AirOS
device = AirOS("10.40.0.130")
device.name
'RM5PomeziaSNode'
device.uptime_tuple
(121, 0, 5)  # a tuple containing device uptime hours, mins and seconds
```

We have just called two simple properties on device, but we can ask device for more specific values or portions of the SNMP tre
    device.next("1.3.6")

Otherwise, if you want simply a value of the tree just type:: device.get_value("oid_you_want_to_ask_for")

4.3.3 OpenWRT example

The same instructions typed above can be applied to OpenWRT itself, just remember to import the correct firmware by typing:

```
from netengine.backends.snmp import OpenWRT
```

4.4 Indices and tables

- genindex
- search