ROCK THE NET

SEW

cmaran

26.09.2014

Content

Job description

Trained competencies:

- Using APIs, Network programming
- Application programming: GUI-programming, parallel programming
- software engineering: buildsystems, testing with mock-objects, design patterns

Basic tasks

Implement a simple-to-use application to monitor and configure a hardware firewall appliance "Juniper NetScreen 5GT ". The firewall allows read access over the SNMP-protocol (your app should be able to test if SNMPv3 is available and if not fallback on SNMPv2c) and write access over Telnet.

Your app should accomplish following tasks:

- List all configured firewall rules (policies) on the device, add the details of the mentioned services and zones as well.
- Allow refreshing of the list by clicking a button and by a configurable time-intervall. Your GUI should remain responsive even with short refresh-intervals!
- Visualize the thru-put for a highlighted firewall-rule (nice2have: multiple rows) in a line-chart (configurable refresh-interval, unit bytes/sec)
- Encapsulate the data retrieval for further reuse and easy expansion. An UML-model of your design will help you defend it at the review!
- Build a visual appealing and easy to use interface (there is more than Swing out there).

Additional information:

- Since there is only one firewall-appliance available, the time each team can test with the hardware will be strictly limited. Therefore it is essentially to use mock-objects to allow testing the app during times where the hardware is not available.
- An additional benefit of using mock-objects will be, that a CI-Server can use them for automated building and testing.
- You only need to consider firewall-rules for TCP and UDP connections in IPv4.
- You can find Information about the SNMP-Mibs special for the manufacturer of the used appliance here (maybe not all of the Mibs work with the used model): http://www.oidview.com/mibs/3224/md-3224-1.html
- For exploring the SNMP-Data coming from the appliance you can use tools like this:

  http://ireasoning.com/mibbrowser.shtml

Effort

Estimated working time

Name	Task	Estimated time
Dall’ Oglio, Krickl, Maran, Schwertberger	Reading the framework’s documentation	4h 00min
Krickl, Maran,	SNMP-Interface implementation	6h 00min
	Display all firewall rules(policies) / Loading data and displaying it in a table	2h 00min
	Refreshing the rules	2h 00min
	Thru-Put Visualization	4h 00min
Dall’Oglio, Schwertberger	GUI: Connect-Page	2h 00min
	GUI: Firewall Monitoring-Page	3h 00min
	Modultesting SNMP-Interface	3h 00min
	Modultesting GUI	1h 30min
	Integrationtesting SNMP-Inface -> GUI	4h 30min
	Systemtest SNMP-Appliance	3h 00min
	Total	**35h 00min**

Required working time

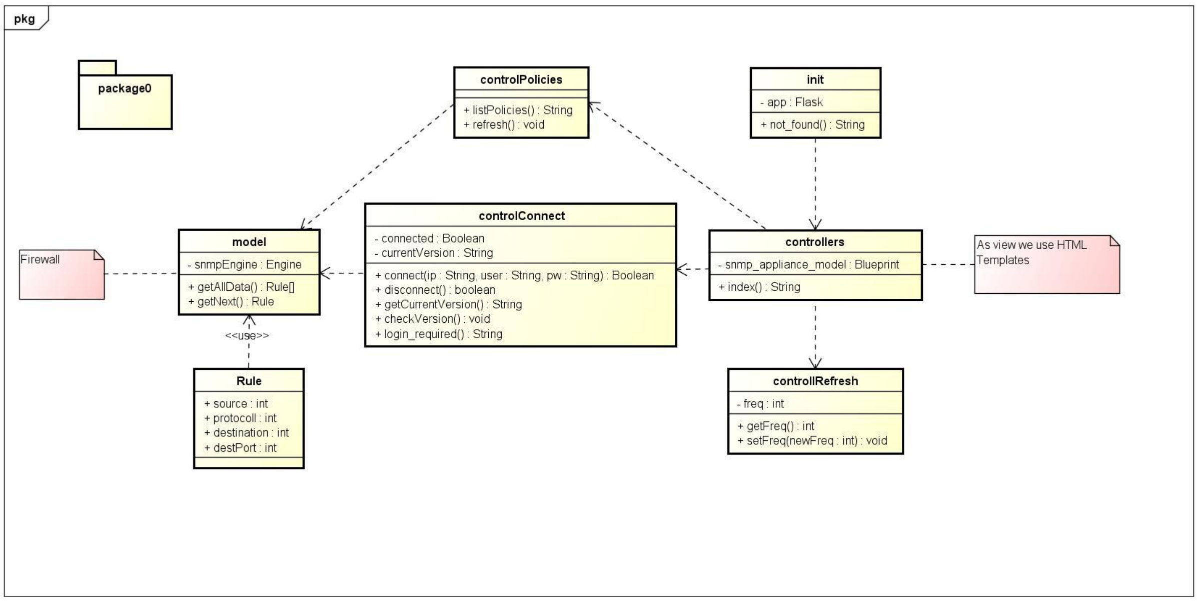
Name	Task	Required time
Dall’ Oglio, Krickl, Maran, Schwertberger	Reading the framework’s documentation	5h 00min
Krickl, Maran,	SNMP-Interface implementation	1h 00min
	Display all firewall rules(policies) / Loading data and displaying it in a table	0h 00min
	Refreshing the rules	0h 00min
	Thru-Put Visualization	0h 00min
Dall’Oglio, Schwertberger	GUI: Connect-Page	1h 00min
	GUI: Firewall Monitoring-Page	1h 00min
	Modultesting SNMP-Interface	1h 00min
	Modultesting GUI	0h 00min
	Integrationtesting SNMP-Inface -> GUI	0h 00min
	Systemtest SNMP-Appliance	0h 00min
	Total	**9h 00min**

Design

For the realization of the tasks we will use Python 3.4.

The backend of the site will be implemented with Flask a Python Web-Framwork where every page is considered as a function. Getting the data from the SNMP-Protocol will be accomplished with the PySNMP framework.

For the frontend Jinja4 will be used to display the data from the Python webservice on the HTML Page. The Traffic-Monitoring could be solved with PyPanl, but we still have to read this framework's documentation. For the GUI-Design we still have to look for some nice look & feel frameworks, like Twitter Bootstrap or Angular.js.For the refreshing of the firewall policies in the GUI we still have to discuss which service we use. By choice we got Server-Sent Events, Websockets, Comet.



Project Layout

/app

- init.py
- controllers.py
- controlRefresh.py
- controlPolicies.py
- model.py
- rule.py
- static → css, fonts, images, js
- templates → HTML Templates

Research

Webframework

- Flask
- Blueprints
    - We thought about using Blueprints to split the system up in **more FILES**

snmp-interface

- PySNMP
    - pySNMP is giving us good examples to retrieve the data from the MIB

monitoring

- PyPANL
    - PyPanl already gives us the possiblity to monitor network transfer

refreshing

- Server-Sent Events
- Websockets
    - Quite a volumnious API so we are little bit scared…
- Comet/APE
    - We don't know if all the stuff is necessary to realize this excerise
- Angular.js
    - Easy to use and an interesting way to solve the problem

responsive-gui

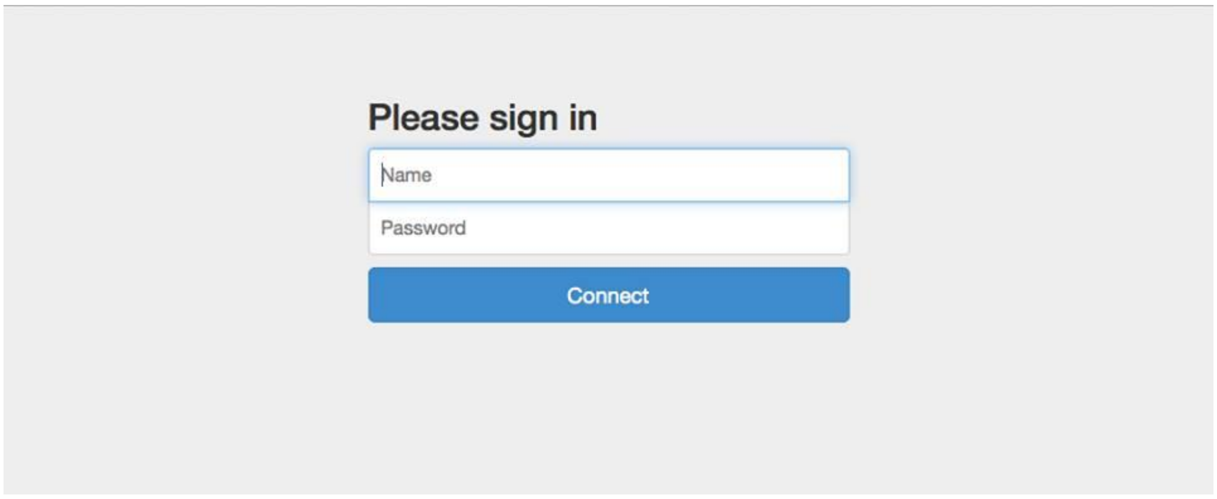
- Twitter Bootstrap - http://getbootstrap.com/

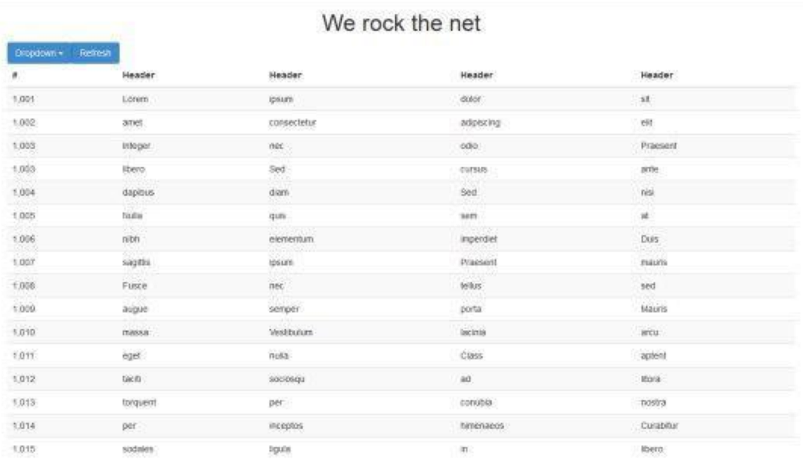
Thru-Put Visualization

- NVD3.js - http://nvd3.org/index.html

Working protocol

GUI-Prototype v1



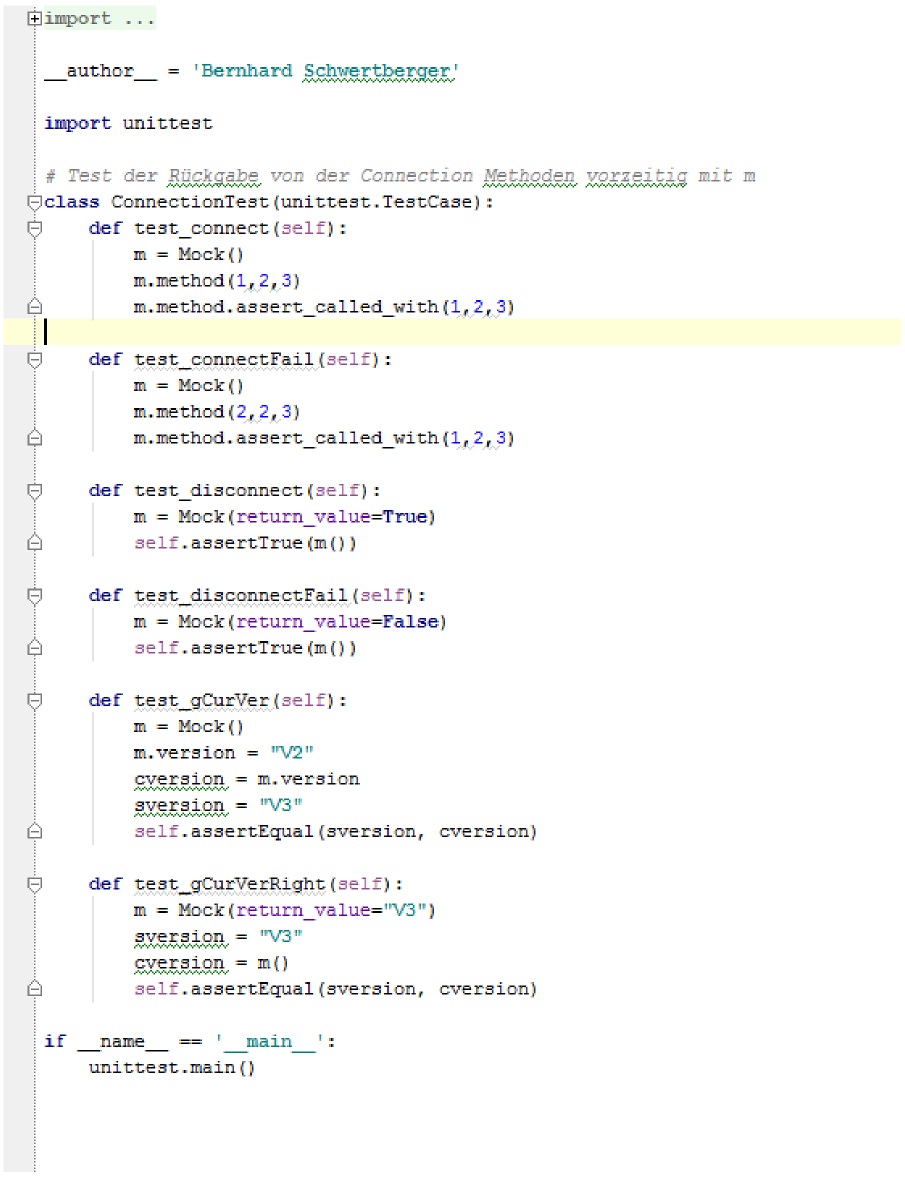


We rock the net

#	Header	Header	Header	Header
1,001	Lorem	ipsum	dolor	sit
1,002	amet	consectetur	adipiscing	elit
1,003	Integer	nec	odio	Praesent
1,003	libero	Sed	cursus	ante
1,004	dapibus	diam	Sed	nisi
1,005	Nulla	quis	sem	at
1,006	nibh	elementum	imperdiet	Duis
1,007	sagittis	ipsum	Praesent	mauris
1,008	Fusce	nec	tellus	sed
1,009	augue	semper	porta	Mauris
1,010	massa	Vestibulum	lacinia	arcu
1,011	eget	nulla	Class	aptent
1,012	taciti	sociosqu	ad	litora
1,013	torquent	per	conubia	nostra
1,014	per	inceptos	himenaeos	Curabitur
1,015	sodales	ligula	in	libero

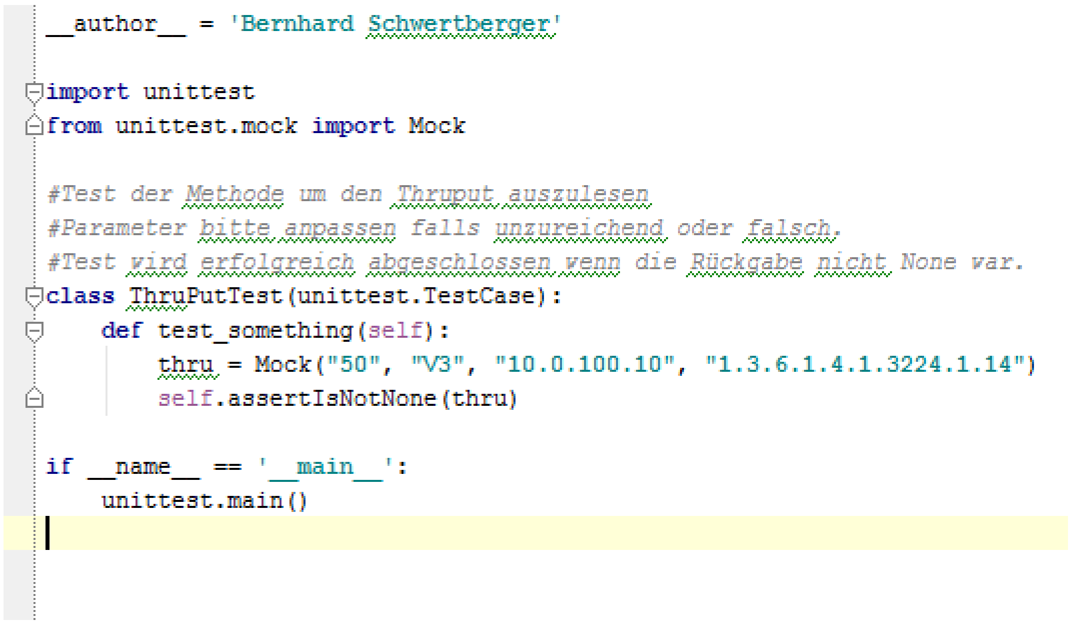
Tests

Connection Test Case



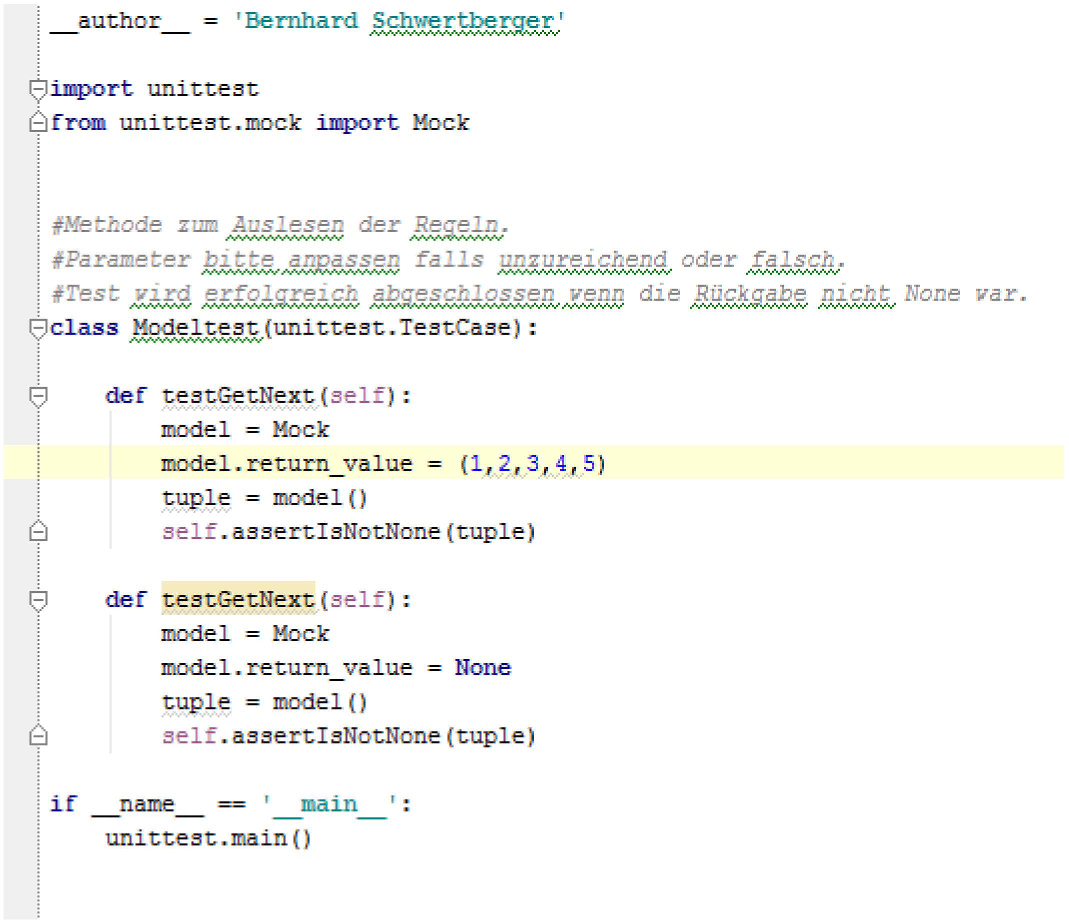
```
import ...

__author__ = 'Bernhard Schwertberger'

import unittest

# Test der Rückgabe von der Connection Methoden vorzeitig mit m
class ConnectionTest(unittest.TestCase):
    def test_connect(self):
        m = Mock()
        m.method(1,2,3)
        m.method.assert_called_with(1,2,3)

    def test_connectFail(self):
        m = Mock()
        m.method(2,2,3)
        m.method.assert_called_with(1,2,3)

    def test_disconnect(self):
        m = Mock(return_value=True)
        self.assertTrue(m())

    def test_disconnectFail(self):
        m = Mock(return_value=False)
        self.assertTrue(m())

    def test_gCurVer(self):
        m = Mock()
        m.version = "V2"
        cversion = m.version
        sversion = "V3"
        self.assertEqual(sversion, cversion)

    def test_gCurVerRight(self):
        m = Mock(return_value="V3")
        sversion = "V3"
        cversion = m()
        self.assertEqual(sversion, cversion)

if __name__ == '__main__':
    unittest.main()
```

ReadThruPut Test Case



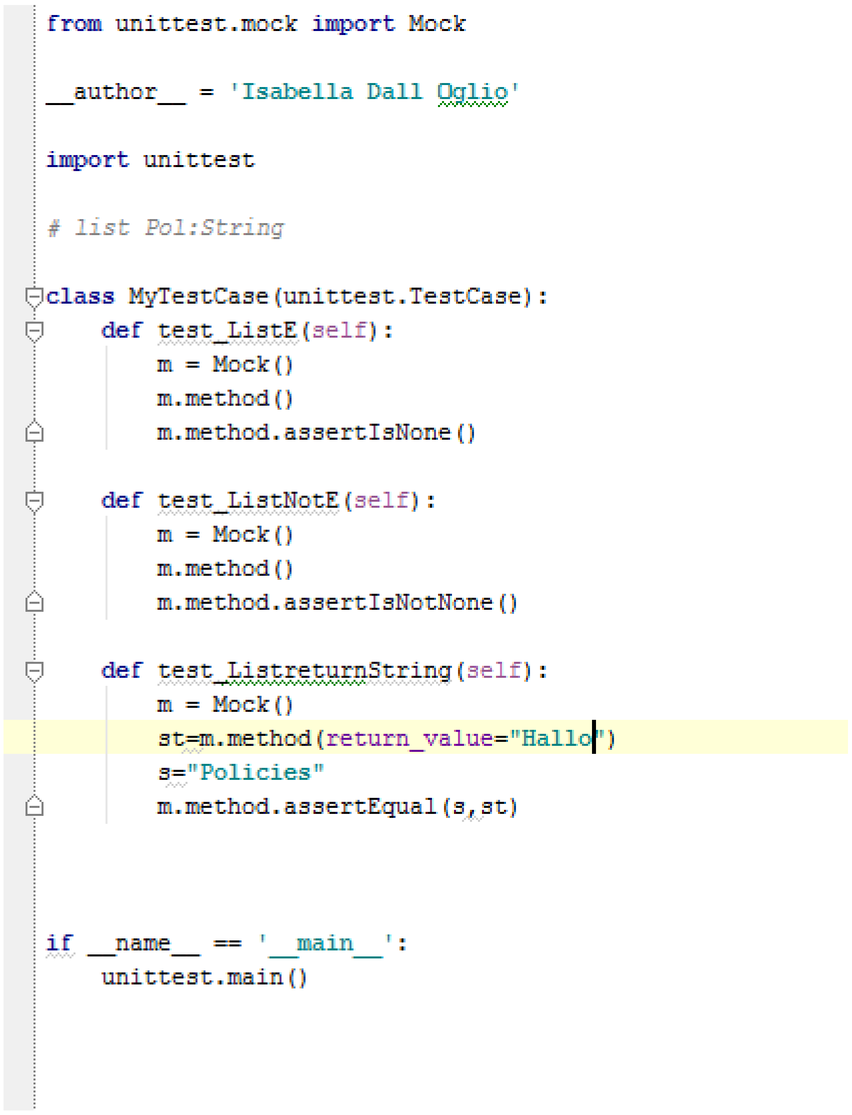
```python
__author__ = 'Bernhard Schwertberger'

import unittest
from unittest.mock import Mock

#Test der Methode um den Thruput auszulesen
#Parameter bitte anpassen falls unzureichend oder falsch.
#Test wird erfolgreich abgeschlossen wenn die Rückgabe nicht None war.
class ThruPutTest(unittest.TestCase):
    def test_something(self):
        thru = Mock("50", "V3", "10.0.100.10", "1.3.6.1.4.1.3224.1.14")
        self.assertIsNotNone(thru)

if __name__ == '__main__':
    unittest.main()
```

Read Test Case



```python
__author__ = 'Bernhard Schwertberger'

import unittest
from unittest.mock import Mock


#Methode zum Auslesen der Regeln.
#Parameter bitte anpassen falls unzureichend oder falsch.
#Test wird erfolgreich abgeschlossen wenn die Rückgabe nicht None war.
class Modeltest(unittest.TestCase):

    def testGetNext(self):
        model = Mock
        model.return_value = (1,2,3,4,5)
        tuple = model()
        self.assertIsNotNone(tuple)

    def testGetNext(self):
        model = Mock
        model.return_value = None
        tuple = model()
        self.assertIsNotNone(tuple)

if __name__ == '__main__':
    unittest.main()
```

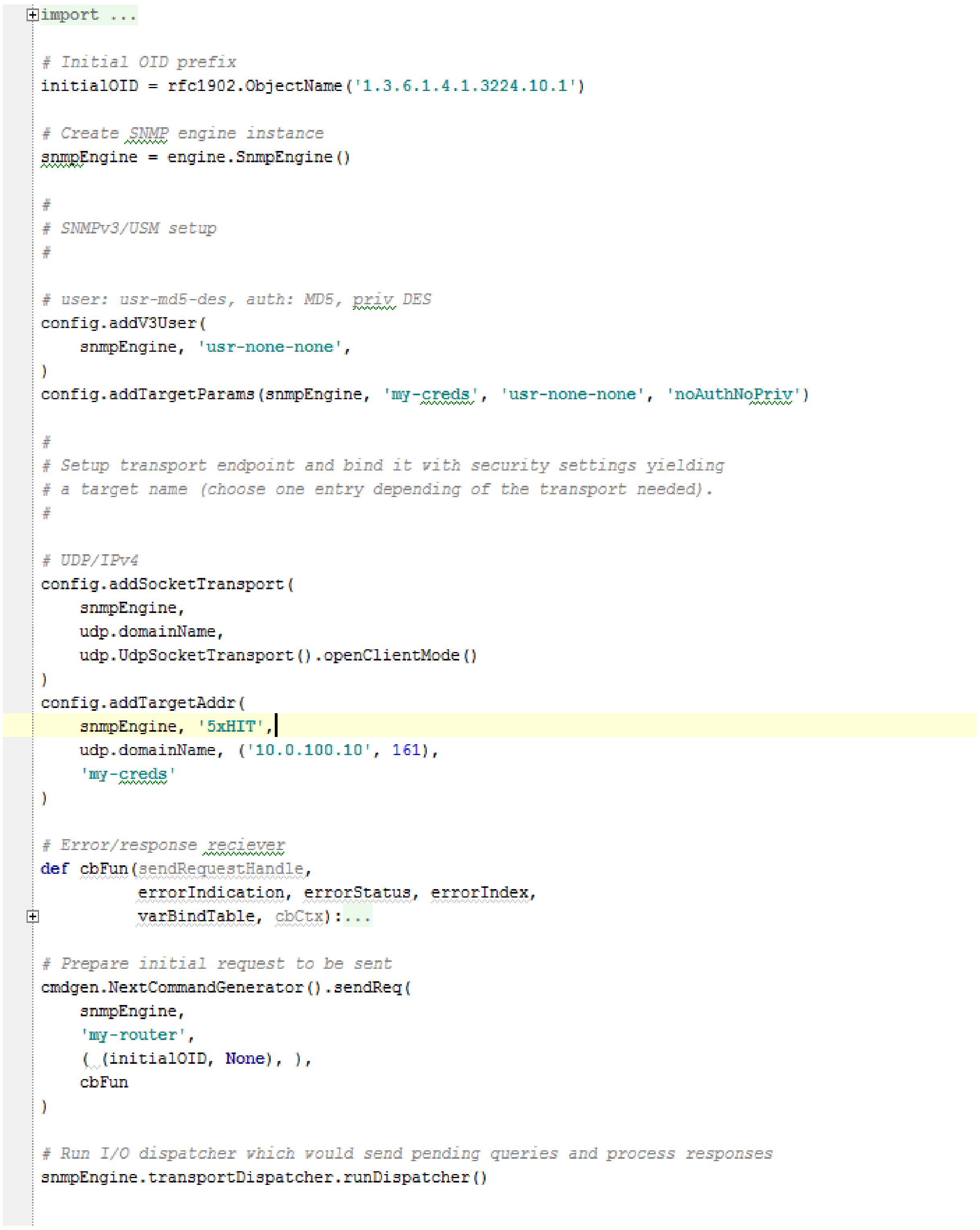
controlPoliciesTest



```
from unittest.mock import Mock

__author__ = 'Isabella Dall Oglio'

import unittest

# list Pol:String

class MyTestCase(unittest.TestCase):
    def test_ListE(self):
        m = Mock()
        m.method()
        m.method.assertIsNone()

    def test_ListNotE(self):
        m = Mock()
        m.method()
        m.method.assertIsNotNone()

    def test_ListreturnString(self):
        m = Mock()
        st=m.method(return_value="Hallo")
        s="Policies"
        m.method.assertEqual(s,st)



if __name__ == '__main__':
    unittest.main()
```

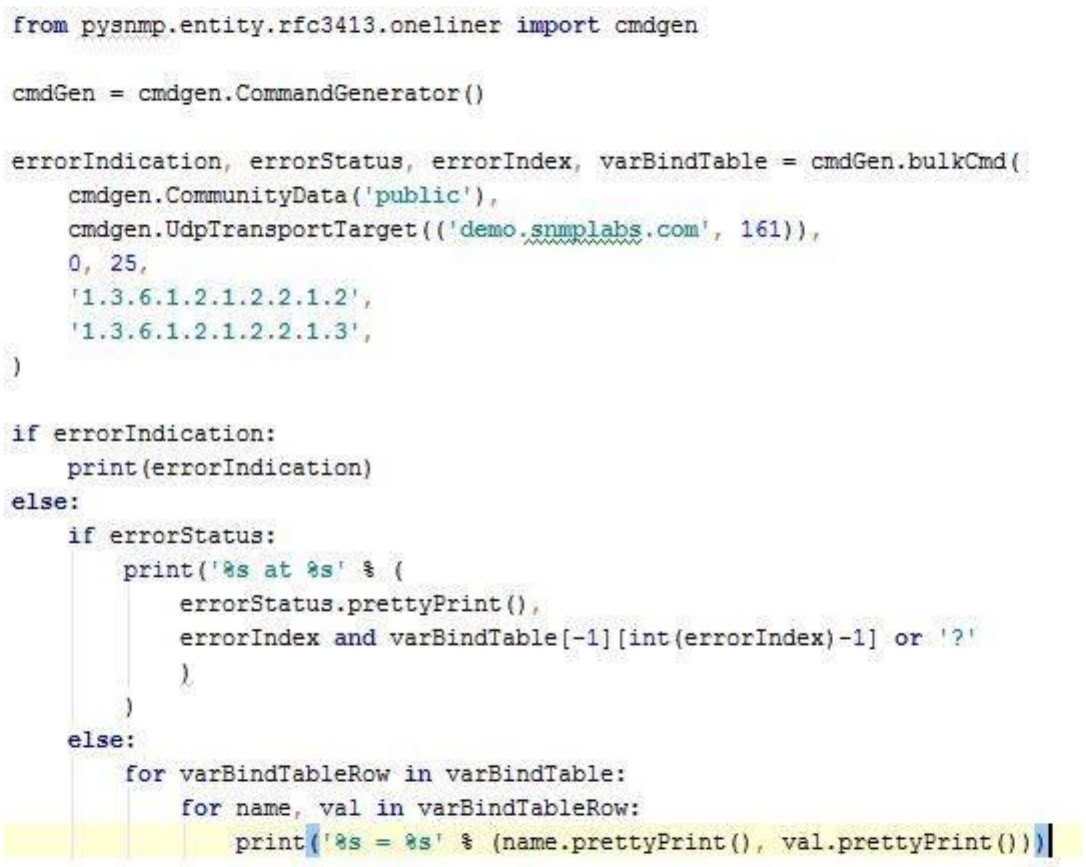
SNMP Appliance

SNMP walk



```
import ...

# Initial OID prefix
initialOID = rfc1902.ObjectName('1.3.6.1.4.1.3224.10.1')

# Create SNMP engine instance
snmpEngine = engine.SnmpEngine()

#
# SNMPv3/USM setup
#

# user: usr-md5-des, auth: MD5, priv DES
config.addV3User(
    snmpEngine, 'usr-none-none',
)
config.addTargetParams(snmpEngine, 'my-creds', 'usr-none-none', 'noAuthNoPriv')

#
# Setup transport endpoint and bind it with security settings yielding
# a target name (choose one entry depending of the transport needed).
#

# UDP/IPv4
config.addSocketTransport(
    snmpEngine,
    udp.domainName,
    udp.UdpSocketTransport().openClientMode()
)
config.addTargetAddr(
    snmpEngine, '5xHIT',
    udp.domainName, ('10.0.100.10', 161),
    'my-creds'
)

# Error/response reciever
def cbFun(sendRequestHandle,
          errorIndication, errorStatus, errorIndex,
          varBindTable, cbCtx):...

# Prepare initial request to be sent
cmdgen.NextCommandGenerator().sendReq(
    snmpEngine,
    'my-router',
    ( (initialOID, None), ),
    cbFun
)

# Run I/O dispatcher which would send pending queries and process responses
snmpEngine.transportDispatcher.runDispatcher()
```

SNMP get



```
from pysnmp.entity.rfc3413.oneliner import cmdgen

cmdGen = cmdgen.CommandGenerator()

errorIndication, errorStatus, errorIndex, varBindTable = cmdGen.bulkCmd(
    cmdgen.CommunityData('public'),
    cmdgen.UdpTransportTarget(('demo.snmplabs.com', 161)),
    0, 25,
    '1.3.6.1.2.1.2.2.1.2',
    '1.3.6.1.2.1.2.2.1.3',
)

if errorIndication:
    print(errorIndication)
else:
    if errorStatus:
        print('%s at %s' % (
            errorStatus.prettyPrint(),
            errorIndex and varBindTable[-1][int(errorIndex)-1] or '?'
            )
        )
    else:
        for varBindTableRow in varBindTable:
            for name, val in varBindTableRow:
                print('%s = %s' % (name.prettyPrint(), val.prettyPrint()))
```

Sources

http://www.w3schools.com/html/html5_serversentevents.asp

http://flask.pocoo.org/

http://flask.pocoo.org/docs/0.10/blueprints/

http://pysnmp.sourceforge.net/

http://blog.panl.com/2011/09/13/using-panl-to-monitor-snmp-enabled-network-devices/

https://ws4py.readthedocs.org/en/latest/

http://ape-project.org/

https://angularjs.org/