**Read me for Asimo web**:

1. Download the latest version of Chrome or Firefox.
2. Download the latest version of Chromedriver (for chrome) or geckodriver (for firefox).

Link to chromedriver: <https://sites.google.com/a/chromium.org/chromedriver/downloads>

Link to geckodriver:

<https://github.com/mozilla/geckodriver/releases>

1. If you are on a mac or a link environment, install selenium (pip install selenium)
2. If you are on a windows environment, download the latest version of selenium from here:

<http://selenium-python.readthedocs.io/installation.html>

(This installs selenium python bindings. Don’t install selenium server. We are only going to be running the selenium client on our machines)

(Ensure that the web drivers are correctly installed in your system path, else they might throw an error)

1. Ensure that you have nose tests installed to run the tests: (pip install nose)
2. Any IDE can be used to develop these test cases. (Pycharm would be the easiest to install and test locally on your system)

**Sample skeleton code is given below:**

Tests for an app id by logging into an application and testing the required functionalities.

(**Imports all the required modules for our code to run**)

import unittest

import time

import re

import utils

import datetime

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.action\_chains import ActionChains

from flaky import flaky

@flaky(max\_runs=3, min\_passes=1,rerun\_filter=utils.delay\_rerun) //decorators to run a test particular no. of times based on passing/failing criteria

class Linkedin(unittest.TestCase): //initialize a class function for the main application, all the sub apps are added as different test cases under this

{

@classmethod

def setUpClass(cls): //Configures the firewall rules for decryption and commits

{

//**code for firewall rules here**

}

def setUp(self): //Configures all the web browser functionalities and also initializes all the urls and login information needed for this particular application

{

//**code for config class here**

}

def test\_01\_linkedin\_like(self): // Selenium code and verification of the firewall session is done here

{

//**code for test case here**

}

} **//class ends**

**Running the tests:**

1. **nosetests -v scriptname**
2. If you are testing for a particular test case under the class, run:

**nosetests -v scriptname:classname.testname**

**Sample code to Login to Linkedin, perform an action (click on the “Like” button) and check the firewall sessions for the associated action (identify “linkedin-posting”).**

@flaky(max\_runs=3, min\_passes=1,rerun\_filter=utils.delay\_rerun)

class Linkedin(unittest.TestCase):

@classmethod

def setUpClass(cls):

‘’’Creates a decryption rule on the firewall and performs commit’’’

utils.configure\_command\_rule(utils.create\_decrypt\_profile)

utils.fw\_commit()

def setUp(self):

'''Configures all the webdriver and web browser options'''

utils.dut\_command(utils.clear\_session) [clears all the firewall sessions before starting the test]

self.wd=utils.func\_chrome\_driver() [configures the chrome browser]

self.base\_url = <https://www.linkedin.com/> [website url]

self.username = [panbot9@gmail.com](mailto:panbot9@gmail.com) [initialize username for login]

self.password = "technsec" [initialize password for login]

def test\_01\_linkedin\_like(self):

'''Tests for linkedin-posting sessions through a "like" action '''

wd = self.wd [initializes the webdriver]

wd.get(self.base\_url) [navigates to the url initialized above]

time.sleep(utils.url\_wait) [waits until the wepage loads]

wd.find\_element\_by\_id("login-email").send\_keys(self.username) [enters the login information]

wd.find\_element\_by\_id("login-password").send\_keys(self.password)

wd.find\_element\_by\_id("login-submit").click() [clicks on the login button]

time.sleep(utils.url\_wait)

wd.find\_element\_by\_xpath("//button[@data-control name='like\_toggle']").click() [clicks on the like button associated with a post on the news feeds]

time.sleep(utils.min\_wait)

result = utils.get\_fw\_sessions() [fetches the firewall sessions]

if "ACTIVE" in result['linkedin-posting']: [checks for the associated session on the firewall]

print "linkedin posting verdict seen in session logs" [passes the test if the app id is found]

else:

self.fail("linkedin posting verdict not found in firewall session logs") [fails the test if app id isn’t detected]

def tearDown(self):

self.wd.quit() [tears down the browser after the test is complete]

@classmethod

def tearDownClass(cls):

‘’’Removes the decryption rule once the test is complete and forces commit’’’

utils.configure\_command\_rule(utils.delete\_decrypt\_profile)

utils.fw\_commit()