

Create Python Scripts To Test Layer 4 Traffic

Goal: Create a script to test Layer 4 traffic using Realm

Using the `realm.py` library we will write a script that will allow us to automate the creation of stations and Layer 4 cross connects. We will also be able to start and stop traffic over the cross connects using the script. Station and Cross Connect creation is covered in the [Realm Scripting Cookbook](#). Requires LANforge 5.4.2.

1.

Creating The Profile

- A. We will use the factory method `self.local_realm.new_l4_cx_profile()` to create our profile object.
- B. After we have done this we can set a few variables for our traffic:
- A. `l4_cx_profile.requests_per_ten` will set our rate of requests per ten minutes. Setting `requests_per_ten = 600` will set our URL request rate to 1 per second. There is no limit to what can be used as the rate but common rates are:
- 600 : 1/s
 - 1200 : 2/s
 - 1800 : 3/s
 - 2400 : 4/s
- B. `l4_cx_profile.url` is the URL to be used in the requests. We will also need to specify the direction (dl/ul) and a absolute path for the destination. See syntax [here](#).

Example:

```
l4_cx_profile.url = "dl http://10.40.0.1 /dev/null"
```

- C. Example Layer 4 profile init:

```
class IPV4L4(LFClientBase):
    def __init__(self, host, port, ssid, security, password, url, requests_per_
        target_requests_per_ten=600, number_template="00000", resource=1, num_t
        _debug_on=False,
        _exit_on_error=False,
        _exit_on_fail=False):
    super().__init__(host, port, _debug=_debug_on, _halt_on_error=_exit_on_err
    self.host = host
    self.port = port
    self.ssid = ssid
    self.security = security
    self.password = password
    self.url = url
    self.requests_per_ten = requests_per_ten
    self.number_template = number_template
    self.sta_list = station_list
    self.resource = resource
    self.num_tests = num_tests
    self.target_requests_per_ten = target_requests_per_ten

    self.local_realm = realm.Realm(lfclient_host=self.host, lfclient_port=self.
    self.cx_profile = self.local_realm.new_l4_cx_profile()
    self.cx_profile.url = self.url
    self.cx_profile.requests_per_ten = self.requests_per_ten

    # Station Profile init
```

2.

Starting Traffic

- A. When running traffic, if you plan to measure the rate of requests, it is recommended to do so in 10 minute increments. An example of this can be seen here: [test_ipv4_l4_urls_per_ten.py](#). To start the traffic we can use the `l4_profile.start_cx()` method. To stop the traffic we can use the `l4_profile.stop_cx()` method.
- B. Example start and build method:

```
def build(self):
    # Build stations
    self.station_profile.use_security(self.security, self.ssid, self.password)
    print("Creating stations")
    self.station_profile.create(resource=1, radio="wiphy0", sta_names_=self.sta_list, c
    temp_sta_list = []
    for station in range(len(self.sta_list)):
        temp_sta_list.append(str(self.resource) + "." + self.sta_list[station])

    self.l4_profile.create(ports=temp_sta_list, sleep_time=.5, debug_=self.debug, supp

def start(self, print_pass=False, print_fail=False):
    temp_stas = self.sta_list.copy()
    temp_stas.append("eth1")
    cur_time = datetime.datetime.now()
    interval_time = cur_time + datetime.timedelta(minutes=1)
    passes = 0
    expected_passes = 0
    self.station_profile.admin_up(1)
    self.local_realm.wait_for_ip(self.resource, temp_stas)
    self.l4_profile.start_cx()
    print("Starting test")
    for test in range(self.num_tests):
        expected_passes += 1
        while cur_time < interval_time:
            time.sleep(1)
            cur_time = datetime.datetime.now()

        if self.l4_profile.check_errors(self.debug):
            if self.__check_request_rate():
                passes += 1
            else:
                self._fail("FAIL: Request rate did not exceed 90% target rate", print_f
                break
            else:
                self._fail("FAIL: Errors found getting to %s " % self.url, print_fail)
                break
            interval_time = cur_time + datetime.timedelta(minutes=1)
    if passes == expected_passes:
        self._pass("PASS: All tests passes", print_pass)
```

3.

Examining The Results

- A. We can use <http://localhost:8080/layer4/list> to check our Layer 4 endpoints. Adding a `,?fields` to the end of the URL will allow us to specify what we want to look at. We can separate fields by commas to show more than one at a time.

Example: <http://localhost:8080/layer4/list?fields=name,urls/s,total-urls>

- Using `total-urls` will show us the total requests made.
- Using `urls/s` will show us the average URL rate per second.
- Using `rx rate` and `tx rate` will show us the rates of received and transeferred traffic.

We can also use the url <http://localhost:8080/layer4/all> to see all of the available fields.

- B. When checking our results for Layer 4 tests we might want to check for common URL related errors:
- `acc. denied` will show us the number of times we got an access denied error.
 - `bad-url` will show us the number of times a request was made with an invalid URL.
 - `nf (4xx)` will count the number of 400 errors recieved when making requests to our URL.

