

Create Python Scripts To Test Generic Traffic

Goal: Create a script to test Generic traffic using Realm

Using the `realm.py` library we will write a script that will allow us to automate the creation of stations and generic 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_generic_cx_profile()` to create our profile object.

B. After we have done this we can set a few variables for our traffic:

A. `gen_cx_profile.type` will determine the type of command to execute.

Example: `self.cx_profile.type = "lfping"`

B. `gen_cx_profile.dest` is the destination IP address for the command.

Example: `self.cx_profile.dest = "127.0.0.1"`

C. `gen_cx_profile.interval` sets the interval at which the command is run in seconds.

Example: `self.cx_profile.interval = 1`

D. Example Generic profile init:

```
class GenTest(LFCLiBase):
    def __init__(self, host, port, ssid, security, password, sta_list, name_pre
        number_template="00000", test_duration="5m", type="lfping", de
        interval=1, radio="wiphy0",
            _debug_on=False,
            _exit_on_error=False,
            _exit_on_fail=False):
        super().__init__(host, port, _debug=_debug_on, _halt_on_error=_exit_on
        self.host = host
        self.port = port
        self.ssid = ssid
        self.radio = radio
        self.upstream = upstream
        self.sta_list = sta_list
        self.security = security
        self.password = password

        self.number_template = number_template
        self.name_prefix = name_prefix
        self.test_duration = test_duration

        self.local_realm = realm.Realm(lfclient_host=self.host, lfclient_port=s
        self.cx_profile = self.local_realm.new_generic_cx_profile()
        self.cx_profile.type = type
        self.cx_profile.dest = dest
        self.cx_profile.interval = interval

# Station Profile init
```

2.

Starting Traffic

A. To start the traffic we can use the `gen_cx_profile.start_cx()` method. To stop the traffic we can use the `gen_cx_profile.stop_cx()` method.

B. Example start and build method:

```
def build(self):
    self.station_profile.use_security(self.security, self.ssid, self.password)
    self.station_profile.set_number_template(self.number_template)
    print("Creating stations")
    self.station_profile.set_command_flag("add_sta", "create_admin_down", 1)
    self.station_profile.set_command_param("set_port", "report_timer", 1500)
    self.station_profile.set_command_flag("set_port", "rpt_timer", 1)
    self.station_profile.create(radio=self.radio, sta_names=self.sta_list, debug=self.debug)
    self.cx_profile.create(ports=self.station_profile.station_names, sleep_time=.5)
    self._pass("PASS: Station build finished")

def start(self, print_pass=False, print_fail=False):
    self.station_profile.admin_up()
    temp_stas = self.sta_list.copy()
    temp_stas.append(self.upstream)
    if self.local_realm.wait_for_ip(temp_stas):
        self._pass("All stations got IPs", print_pass)
    else:
        self._fail("Stations failed to get IPs", print_fail)
        exit(1)
    cur_time = datetime.datetime.now()
    passes = 0
    expected_passes = 0
    self.cx_profile.start_cx()
    time.sleep(15)
    end_time = self.local_realm.parse_time("30s") + cur_time
    print("Starting Test...")
    while cur_time < end_time:
        cur_time = datetime.datetime.now()
        gen_results = self.json_get("generic/list?fields=name,last+results", debug=self.debug)
        if gen_results['endpoints'] is not None:
            for name in gen_results['endpoints']:
                for k, v in name.items():
                    if v['name'] in self.cx_profile.created_endp and not v['name'].endswith('...'):
                        expected_passes += 1
                        if v['last results'] != "" and "Unreachable" not in v['last results']:
                            passes += 1
                    else:
                        self._fail("%s Failed to ping %s " % (v['name'], self.cx_profile.station_names[k]), print_fail)
                        break
        time.sleep(1)
    if passes == expected_passes:
        self._pass("PASS: All tests passed", print_pass)
```

3.

Examining The Results

- A. For **lfping** we can use the last results of the endpoint to determine if the test was successful. An example of this can be seen in our **start** method. The most common errors for **lfping** will either be a blank last result or **Destination Host Unreachable**. Either of these results indicate a failed ping. Successful pings will look like:

```
64 bytes from 10.40.0.1: icmp_seq=1 time=4.55 ms *** drop: 0 (0, 0.000) rx: 1 fail: 0
```

Results can also be seen in the generic tab in the LANForge Manager:

The screenshot shows the LANForge Manager interface. At the top, there are tabs for Control, Reporting, Windows, Info, and Tests. Below these are various management buttons like Chamber View, Stop All, Restart Manager, Refresh, and HELP. A navigation bar includes tabs for Generic, Resource Mgr, vAP Stations, DUT, Profiles, Traffic-Profiles, Alerts, Messages, Warnings, Wifi-Messages, Status, Port Mgr, Layer-3, L3 Endps, Layer 4-7, Armageddon, WanLinks, VoIP/RTP, VoIP/RTP Endps, and File-IO. A Rpt Timer is set to 'fast (1 s)' and the Test Manager is set to 'all'. Below this is a table titled 'Generic Endpoints for Selected Test Manager'.

Name	EID	Status	Rpt#	Last Results	Tx Bytes	Rx Bytes	Tx Pkts	PDU/s TX	Rx F
sta0000_gen0	1.1.13.5	Stop...	206	64 bytes from 10.40.0.1: icmp_seq=204 time=3.75 ms *** dr...	0 B	0 B	0	0	0
sta0001_gen0	1.1.14.7	Stop...	206	64 bytes from 10.40.0.1: icmp_seq=204 time=1.61 ms *** dr...	0 B	0 B	0	0	0

Logged in to: 192.168.92.12:4002 as: Admin

Double-clicking on an endpoint will allow you to see more specific results as well as the command used by the endpoint. Using the **sync** button will allow you to see updated results.

The screenshot shows the 'Create/Modify Generic Endpoint' dialog box. It contains fields for Name (sta0000_gen0), Rpt Timer (default (5 s)), Test Manager (default_tm), Shelf (1), Resource (1 (ct524-emily)), Port (13 (sta0000)), and Endp ID (5). The Command Builders dropdown is set to 'ping'. Below this are settings for Size (Default), Interval (1000000 (1 s)), and Count (Infinite). The Target field contains '10.40.0.1' and the Payload field is empty. The Command field shows 'lfping -i 1 -I sta0000 10.40.0.1'. The Command Output section displays a list of ping results for sequence numbers 167 through 264, all showing successful responses with varying times and 0 drops. At the bottom, there are buttons for Sync, Apply, OK, and Cancel, with the Sync button highlighted by a red box.