

Project Report
On
Lab Evaluation Monitoring Tool
By Group 1018

Name of the student

ID No.

Raunak Ritesh

2015A7PS0160H

Sanyam Jain

2015A7PS0094H

Geetanjali Dhakar

2015A3PS0372H

During the course of
Network Programming (IS F462)

Under the guidance of
Dr. Paresh Saxena



BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(1st February - 8th April, 2019)

Table of Contents

1. Introduction
2. Project Design
3. Implementation Steps:
 - a. Server Code
 - b. Client Code
 - c. Libraries Used
4. Additional Features
5. Code Submission and How to Run

PROJECT INTRODUCTION

The objective of this project is to develop a monitoring tool to stop copying during a lab evaluation. The project has 3 major capabilities:

1. To Detect a connected/removed USB.
2. To Detect Packets traversing for Internet.
3. To Detect Packets traversing for LAN.

Once the lab evaluation is completed, all these logs are transferred to the server in the lab. The IP address of the server is fixed and known by all the other PCs. The application shall be configured such that it is always running in the background and stores logs only for the given period.

PROJECT DESIGN

On the Client side, we've implemented the logging system in real-time, i.e. an API hit is done the moment any of the above connection is detected. Along with the request, the report is also saved in a local file at the client PC.

For the Server side, we've used Django to host a server, service multiple clients at the same time, store logs of each client in a SQLite Database and provide a web interface for the administrators.

Apart from the Server and the Client Side, the project uses Python 2.7, Django, PyUdev (USB Monitoring), Scapy(Packet Monitoring), Requests and the Socket Library.

Implementation Steps - SERVER CODE

1. How does the server look on running?

The following command is used to run the server locally and yet have it accessible by all its clients on LAN.

Command - **python manage.py runserver <IP>:8000**

```
(network-monitoring-tool) raunak@raunak-HP-Pavilion-15-Notebook-PC:/media/raunak/Education/BITS/CSE/Projects/network-monitoring-tool/NetMonitorTool$ python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
April 07, 2019 - 11:45:50
Django version 1.11.20, using settings 'NetMonitorTool.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

2. How to store each Client's data on the server side?

With the help of models, which act as the database definition on the server side, we can define tables (classes) and attributes under each table.

```
models.py
1  # -*- coding: utf-8 -*-
2  from __future__ import unicode_literals
3
4  from django.db import models
5
6  # username - admin
7  # p/w - administrator
8
9  # Create your models here.
10 class LabReport(models.Model):
11     clientIP = models.CharField(max_length = 15, primary_key = True)
12     clientName = models.CharField(max_length = 100, blank = False)
13     clientStatus = models.BooleanField(default = False)
14     flag = models.BooleanField(default = False)
15
16     usbDetected = models.BooleanField(default = False)
17     # usbTimestamp = models.DateField(null = True)
18
19     internetDetected = models.BooleanField(default = False)
20     # internetTimestamp = models.DateField(null = True)
21
22     lanDetected = models.BooleanField(default = False)
23     # lanTimestamp = models.DateField(null = True)
24
25     def __str__(self):
26         return '%s' % (self.clientIP)
27
28     class Meta:
29         ordering = ['flag']
30
```

3. Which Database is used by the Server?

A sqlite file is stored on the backend which acts as the database.

```
# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

4. How is a request handled by the Server?

It can be seen that a POST function is defined on the server side. Each time a client makes a request to the server, data is saved on the server in the sqlite DB which is further accessed on the Admin Dashboard

```
class Logs(View):
    def post(self, request):
        hostname = str(request.POST.get('hostname', ''))
        ip = str(request.POST.get('ip', ''))
        flag = int(request.POST.get('flag', ''))
        booleanStatus = request.POST.get('booleanStatus', '')

        print(ip + ' ' + str(flag) + ' ' + booleanStatus)

        if flag == 0: #Up and Running. Create New Entry
            entry = LabReport(clientIP = ip,\
                               clientName = hostname,\
                               clientStatus = booleanStatus,\
                               flag = False,\
                               usbDetected = False,\
                               internetDetected = False,\
                               lanDetected = False)
            entry.save()
        elif flag == 1: # USB Flag
            entry = LabReport.objects.get(clientIP = ip)
            entry.usbDetected = booleanStatus
            entry.flag = booleanStatus
            entry.save()
        elif flag == 2: # Internet and LAN Flag
            entry = LabReport.objects.get(clientIP = ip)
            entry.internetDetected = booleanStatus
            entry.lanDetected = booleanStatus
            entry.flag = booleanStatus
            entry.save()

        # with open(BASE_DIR + '/../LabReport.txt', 'a+') as outfile:
        #     outfile.write(json.dumps(info, sort_keys = False, indent = 4))

        return HttpResponseRedirect('Info Saved Successfully', status = 200)
```

5. How is Data accessed on the Admin Dashboard?

The report can be accessed on the Server PC, through the following link:

Link : <http://localhost:8000/admin/LabMonitor/labreport/>

For each Client, an entry is made and the boolean status shows which all connections have been observed on the client PC in real-time.

Django administration

WELCOME, ADMIN.

Home > Labmonitor > Lab reports

Select lab report to change

Action:

Go

 0 of 1 selected

<input type="checkbox"/>	CLIENTIP	CLIENTNAME	CLIENTSTATUS	FLAG	USBDETECTED	INTERNETDETECTED	LANDETECTED
<input type="checkbox"/>	127.0.1.1	raunak-HP-Pavilion-15-Notebook-PC	✓	✓	✓	✗	✗

1 lab report

Implementation Steps - CLIENT CODE

USB - MONITOR

1. *What is the logic used behind USB port Monitoring?*

The function takes in a parameter 'Context' which is an object of type pyudev. The pyudev Context can be further used to emit events whenever a device is connected or removed. With the help of Monitor, we can log such events synchronously for any given period of time (Line 20 - 7200 seconds).

```
13 def synchronousMonitoring(context):
14     try:
15         # Creates a Monitor which keeps on running
16         monitor = pyudev.Monitor.from_netlink(context)
17         monitor.filter_by('block')
18
19         # Runs for the next 2 hours once an event is logged
20         for device in iter(partial(monitor.poll, 7200), None):
21             msgstring = '{0.action} on {1}'.format(device, device.get('ID_FS_LABEL'))
22             print(msgstring)
23
24             if device.action == 'add' or device.action == 'remove':
25                 postLogsToServer(1, True)
26
27     except Exception, e:
28         postLogsToServer(0, False)
29
30     print(e)
31     synchronousMonitoring(context)
```

2. *What is the library 'Pyudev' and how does it monitor USB Ports?*

Pyudev is the device and hardware management library of Linux. But before we can monitor anything, a connection needs to be established to the device database first. This connection is represented by the library Context() which is passed to the usbMonitoring Function.

```
context = pyudev.Context()
synchronousMonitoring(context)
```

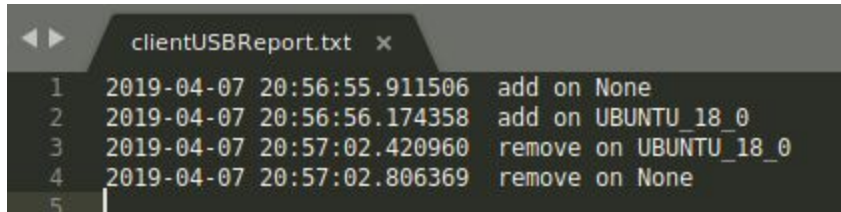
3. *What happens when a USB is connected or removed?*

The add or remove shows the action taken by a device.

```
(network-monitoring-tool) raunak@raunak-HP-Pavilion-15-Notebook-PC: /media/raunak/Education/BITS/C
SE/Projects/network-monitoring-tool/NetMonitorTool/CLIENT-CODE$ python usbMonitor.py
add on None
add on UBUNTU_18_0
```


4. *What is the logic behind logging at Client's Side?*

The following screenshot shows a snippet of the report generated on the client' side.



1	2019-04-07 20:56:55.911506	add on None
2	2019-04-07 20:56:56.174358	add on UBUNTU_18_0
3	2019-04-07 20:57:02.420960	remove on UBUNTU_18_0
4	2019-04-07 20:57:02.806369	remove on None
5		

5. *How is a request sent to the server in Real-Time if and when an USB is detected?*

The following function shows how a POST request is sent to the Server whenever an USB Connection is detected. At line 44, it can be seen that at the specified fixed url, a dictionary is sent having the hostname, ip.

```
33 def postLogsToServer(flag, booleanStatus):
34     hostname, ip = getSystemInfo()
35     url = "http://localhost:8000/recordlogs/"
36
37     params = {
38         'hostname' : hostname,
39         'ip'       : ip,
40         'flag': flag,
41         'booleanStatus' : booleanStatus
42     }
43
44     r = requests.post(url = url, data = params)
45
46 def getSystemInfo():
47     hostname = socket.gethostname()
48     ip = socket.gethostbyname(hostname)
49     return (hostname, ip)
```

On the server side, the same request is shown when received.

```
127.0.1.1 0 True
[07/Apr/2019 11:47:14] "POST /recordlogs/ HTTP/1.1" 200 23
127.0.1.1 1 True
[07/Apr/2019 11:47:21] "POST /recordlogs/ HTTP/1.1" 200 23
127.0.1.1 1 True
[07/Apr/2019 11:47:21] "POST /recordlogs/ HTTP/1.1" 200 23
```


PACKET - MONITOR

1. *What is the logic used behind Internet and Lan Monitoring?*

Scapy comes with an inbuilt packet sniffing function 'sniff' with attributes which define actions on captured packets.

'Prn' - What action to take on each captured packet.

'Stop_filter' - After each capture, the given function decides whether to continue or stop.

'Count' - Number of packets to capture, currently set to unspecified.

```
23 start = time.time()
24
25 sniff(filter="ip",\
26       # prn=lambda x:x.sprintf("{IP:%IP.src% -> %IP.dst%\n}")
27       prn = mainfunc,
28       count = 0,
29       stop_filter = timer)
30
31 def timer(x):
32     if time.time() - start >= 1000:
33         return True
34     else:
35         return False
36
37 def mainfunc(x):
38     print(x.summary())
39     if x.haslayer(TCP):
40         postLogsToServer(2, True)
41
42     return True
```

The captured packet summary is shown below:

```
-tool/NetMonitorTool/CLIENT-CODE$ sudo python packetMonitor.py
[sudo] password for raunak:
Ether / IP / UDP 172.16.46.154:54915 > 172.16.47.255:54915 / Raw
True
Ether / IP / UDP 172.16.46.154:54915 > 172.16.47.255:54915 / Raw
True
Ether / IP / UDP 172.16.46.154:54915 > 172.16.47.255:54915 / Raw
True
Ether / IP / UDP 172.16.46.154:54915 > 172.16.47.255:54915 / Raw
True
Ether / IP / UDP 172.16.46.154:54915 > 172.16.47.255:54915 / Raw
```

2. *What is Scapy and what all alternatives are there for packet sniffing?*

Scapy is a Python program that enables the user to send, sniff and dissect and forge network packets. This capability allows construction of tools that can probe, scan or attack networks.

3. *How is a request sent to the server in Real-Time if and when a connection is detected?*

The same logic from USB Monitor is borrowed here also, by making an API Hit in real-time.

LIBRARIES USED

1. Django

- a. To host a server
- b. Provide threading capabilities to service multiple clients.
- c. To provide a stable database framework on the Server Side

2. Socket

- a. To get the hostname and ip address of a client.

3. Requests

- a. To make a POST request to the Server from a client.

4. Pyudev

- a. Provides the Linux device and hardware management library for USB Monitoring

5. Scapy

- a. A Stable framework with an in-built sniffer function to capture packets and take further actions.

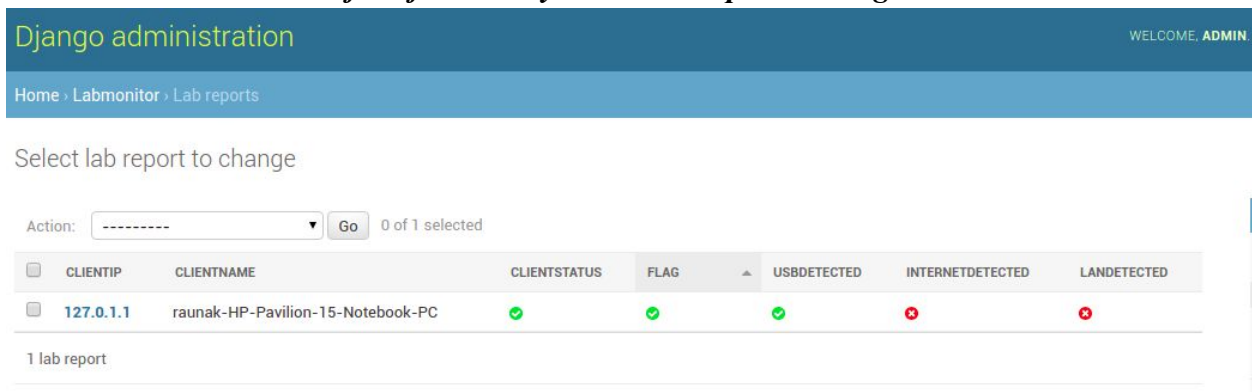
ADDITIONAL FEATURES

1. *Sound Notification at the Server Side*

```
> os.system('spd-say "ALERT"')
```

The following command speaks 'ALERT' each time an event is recorded.

2. *Web Browser Interface for an easy and clean report viewing.*



Django administration

WELCOME, ADMIN.

Home · Labmonitor · Lab reports

Select lab report to change

Action: Go 0 of 1 selected

CLIENTIP	CLIENTNAME	CLIENTSTATUS	FLAG	USBDETECTED	INTERNETDETECTED	LANDETECTED
127.0.1.1	raunak-HP-Pavilion-15-Notebook-PC	✓	✓	✓	✗	✗

1 lab report

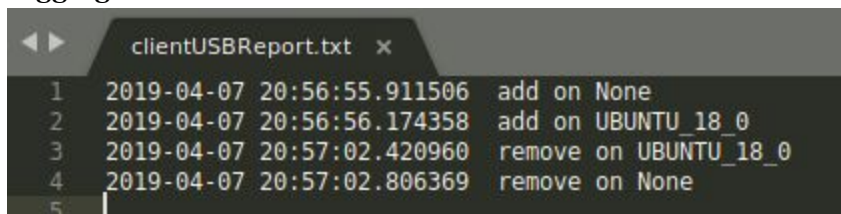
3. *Real-time Events are recorded and sent to the server.*

```
> url = "http://localhost:8000/recordlogs/"
```

```
> r = requests.post(url = url, data = params)
```

The above commands are used to log an events in real-time at the server.

4. *Logging is done at both the Server and Client Level.*



```
clientUSBReport.txt x
1 2019-04-07 20:56:55.911506 add on None
2 2019-04-07 20:56:56.174358 add on UBUNTU_18_0
3 2019-04-07 20:57:02.420960 remove on UBUNTU_18_0
4 2019-04-07 20:57:02.806369 remove on None
5
```

5. *Database Support is provided at the Server side.*



```
# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Code Submission and How to Run

Link - [Code Repository](#)

Github Link - [Code@Github](#)

For the server side

```
> virtualenv .  
> pip install -r requirements.txt  
> python manage.py runserver <IP>:<Port Number>
```

On the Client Side

```
> virtualenv .  
> pip install -r requirements.txt  
> python usbMonitor.py  
> sudo python packetMonitor.py
```