Project Report:

Distributed Network and

TCP Port Scanner with Web UI

Ishan Mehta (109596970)
Naman Mittal (109888028)
Shrey Shah (109247887)
Tanvi Sirsat (110098393)

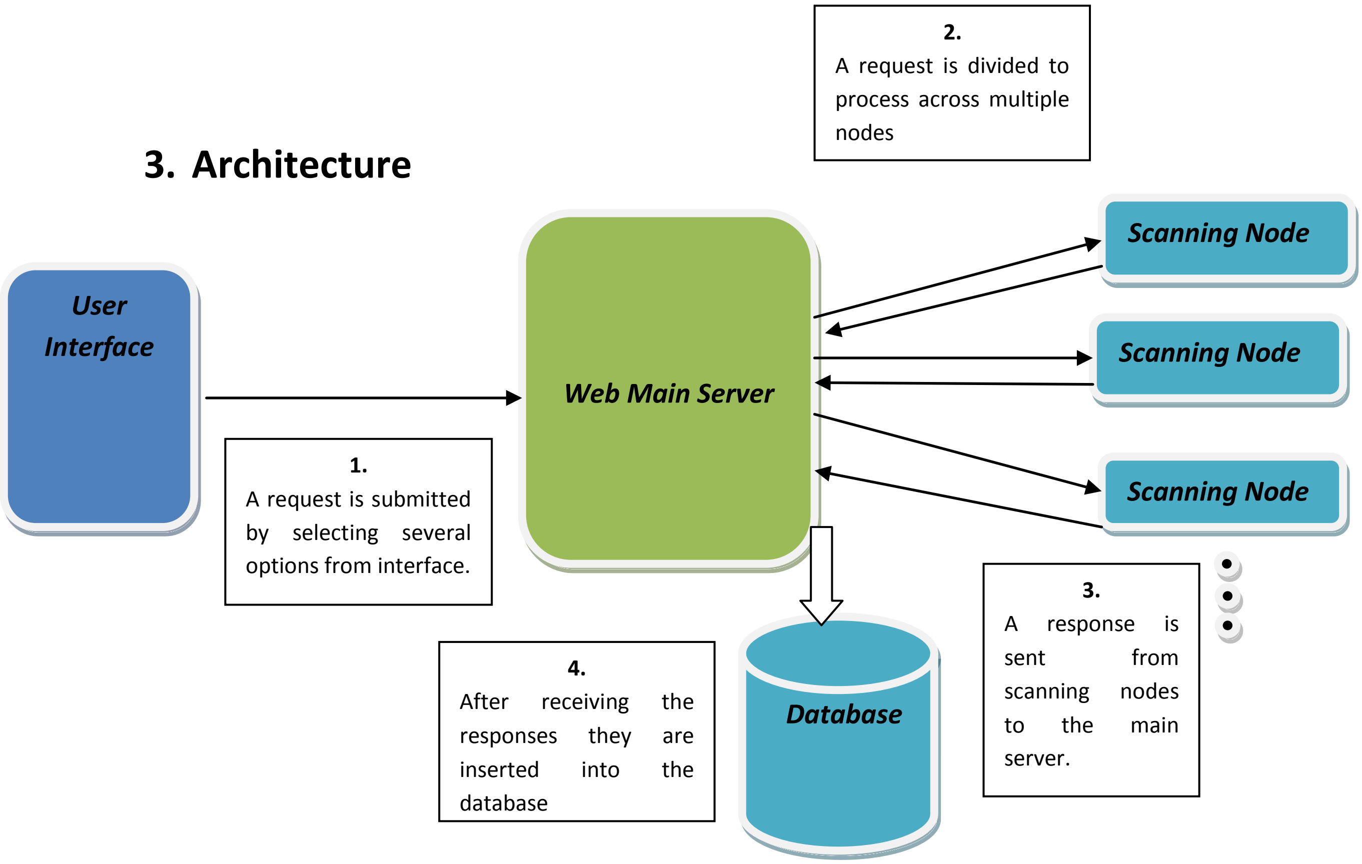
1. Introduction

Network Scanners and Port Scanners are essential tools when trying to understand the layout of a network and the services that a specific host is running. We have implemented a Distributed Network and Port Scanner with a Web UI that can be used to control the scanner. The scanner is distributed across multiple clients so that many hosts (called scanning nodes) can participate in a single scanning effort.

2. Features

- The system supports individual host scan, multiple hosts scan and host scan over a range of ports
- The system supports 3 types of port scanning mode: Normal Port Scanning (full TCP connect to the remote IP address) ,TCP SYN Scanning , TCP FIN Scanning.
- The request can be given with a random flag on so the multiple hosts or port scan does not happen sequentially.
- All the results are stored in the database and can be retrieved and also can be search by giving a specific date.

3. Architecture



2.
A request is divided to process across multiple nodes

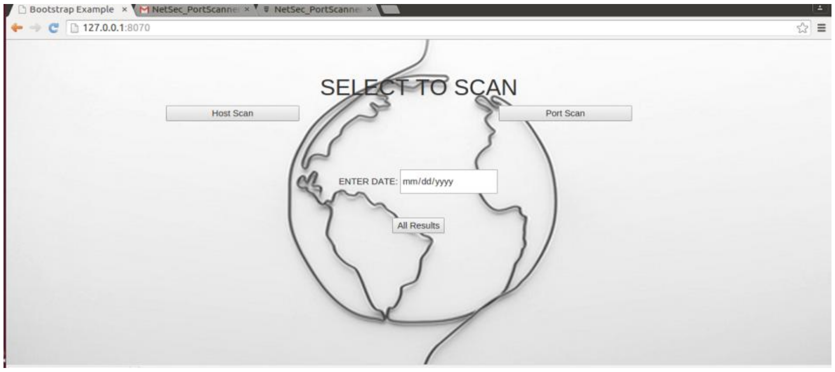
1.
A request is submitted by selecting several options from interface.

3.
A response is sent from scanning nodes to the main server.

4.
After receiving the responses they are inserted into the database

4. General Flow of the implementation

- A request is submitted from the GUI using several options.



- Once the request is submitted the server determines what the type of request is. The system supports 3 types of request. First is individual host scan, second is multiple host scan and third is multiple ports for a single host scan.
- According to the request the number of scans perform are divided equally amongst the scanning nodes which are clients and the individual clients request is sent.
- Both server and scanning nodes support the mechanism of producer consumer queue to handle requests asynchronously.
- The requests are processed and the response packet containing the results is sent back to the main server.
- The server after receiving the responses inserts them into the database.
- The user can then view the results obtained.

5. Implementation Notes

- GUI is implemented using HTML/Javascript and jQuery
- The web server is implemented in Python
- The distributed architecture is implemented using TCP network programming in Python
- The scanning nodes use the scapy library in python.
- Database is implemented using SQLLite3.

6. Database Tables

Following are the tables created in the database to store the requests and the responses:

1. IPINFO is the main table in which all submitted requests are kept into the database.

```
CREATE TABLE IPINFO
(
        IP TEXT,
        TYPE INTEGER,
        ALIVE BOOLEAN,
        TIME DATETIME,
            DATE1 DATE,
            LENGTH INTEGER,
            PRIMARY KEY(IP,TIME)
)
```

IPINFO

IP	TYPE	ALIVE	TIME	DATE1	LENGTH

- IP – Ip address
- Type – Type of Request – 1 – Host Scan 2- Multiple Host Scan 3 – Multiple Port Scan
- ALIVE – Flag set if the IP is alive
- TIME – time stamp of the request
- DATE1- date of the request
- LENGTH – No of ports or no of hosts for type 2 or 3 respectively.

2. PORTDATA is the table which holds data of the responses of the port scan requests.

```
CREATE TABLE PORTDATA
(
            PORT INTEGER,
            IP TEXT,
            TIME DATETIME,
            ALIVE BOOLEAN,
            DATE1 DATE,
            FOREIGN KEY(IP,TIME) REFERENCES IPINFO(IP,TIME)
)
```

PORTDATA

PORT	IP	TIME	ALIVE	DATE1

- PORT – Port number
- IP – Ip address
- TIME – Time stamp of the request
- Alive – Set to tru if the port is alive
- DATE1 – Date of the request

3. IPDATA is the table which holds data of the responses of multiple hosts scan.

```
CREATE TABLE IPDATA
(
            IP TEXT,
            ALIVE BOOLEAN,
            TIME DATETIME,
            DATE1 DATE,
            FOREIGN KEY(IP,TIME) REFERENCES IPINFO(IP,TIME)
```

)

IPDATA

IP	ALIVE	TIME	DATE1

IP – Ip address
ALIVE – Set if the IP address is alive
TIME – time stamp of the request
DATE – date of the request

7. Important functions description

Server.py

- **def add_client(): -**
  This function is to add new scanning nodes and to store the client address in the list handled by the server

- **def client_listen():**
  This function is listening on a specific port for new scanning node to be added

- **def send_client(client_address, request,start, end, ip_list):**
  This function is used to send the request to scanning nodes

- **class ConsumerResponseThread(Thread):**
  def ProducerResponse(response):
  This is producer consumer queue for the responses obtained from the clients.

- **class ConsumerThread(Thread):**
  def Producer (request):
  This is producer consumer queue for the request sent to the client.

client_code.py

- **def scan_ip(ip_list):**
  This functions scans a list of ip's given to it.

- **def scan_port_ack(port_list,ip_addr,logger):**
  This function takes in a list of ports and scans using SYN scanning.

- **def scan_port_connect(port_list,ip_addr,logger):**
  This function takes in a list of ports and scans using full connect scanning.

- **def scan_port_fin(port_list,ip_addr,logger):**
  This function takes in a list of ports and scans using FIN scanning.

8. Testing

Following scenarios have been tested

- Checked if the single host is alive and check if the result is displayed correctly in UI.
- Checked if multiple host scan is working with and without random flag set and all the host results are correctly displayed
- Checked if the multiple port scan is working with and wothout random flag set and all the port results are correctly displayed.
- Checked if the requests are getting properly divided among all the scanning nodes
- Checked if the results can be obtained by searching date.
- Checked if all the three scanning modes are working.

9. Instructions :

NOTE: Please change your internet connection to Wolfie-Guest. (Wolfie-Secure/Open blocks the ICMP packet which is required for pinging the server).

STEPS TO RUN THIS PROJECT:

- STEP 1: Install Scapy and NetAddr packages for python (Scapy : http://www.secdev.org/projects/scapy/ || NetAddr : https://pypi.python.org/pypi/netaddr)
- STEP 2: On machine which you will like to be a server run portscanner_db.py
  (Eg: python portscanner_db.py)
- STEP 3: Initialize server by running in Server.py
  (Eg: python Server.py)
- STEP 4: On Client machines start client by running client_code.py using sudo command or being a root with a command line argument of a port number. For eg "sudo python client_code.py 10005"
- STEP 5: You are set to go you can access service on port 8070 (eg: 127.0.0.1:8070)

10.Resources for reference

- https://docs.python.org/2/howto/sockets.html - Socket Programming
- http://pymotw.com/2/socket/tcp.html - To develop the Client Server communication
- http://www.pythonforpentesting.com/2013/10/port-scanning-with-python.html - Banner Grabbing
- https://securitylair.wordpress.com/2014/02/21/simple-port-scanner-in-python-with-scapy-2/ - Scapy Implementation
- http://agiliq.com/blog/2013/10/producer-consumer-problem-in-python/ - Producer Consumer Problem