Tasks

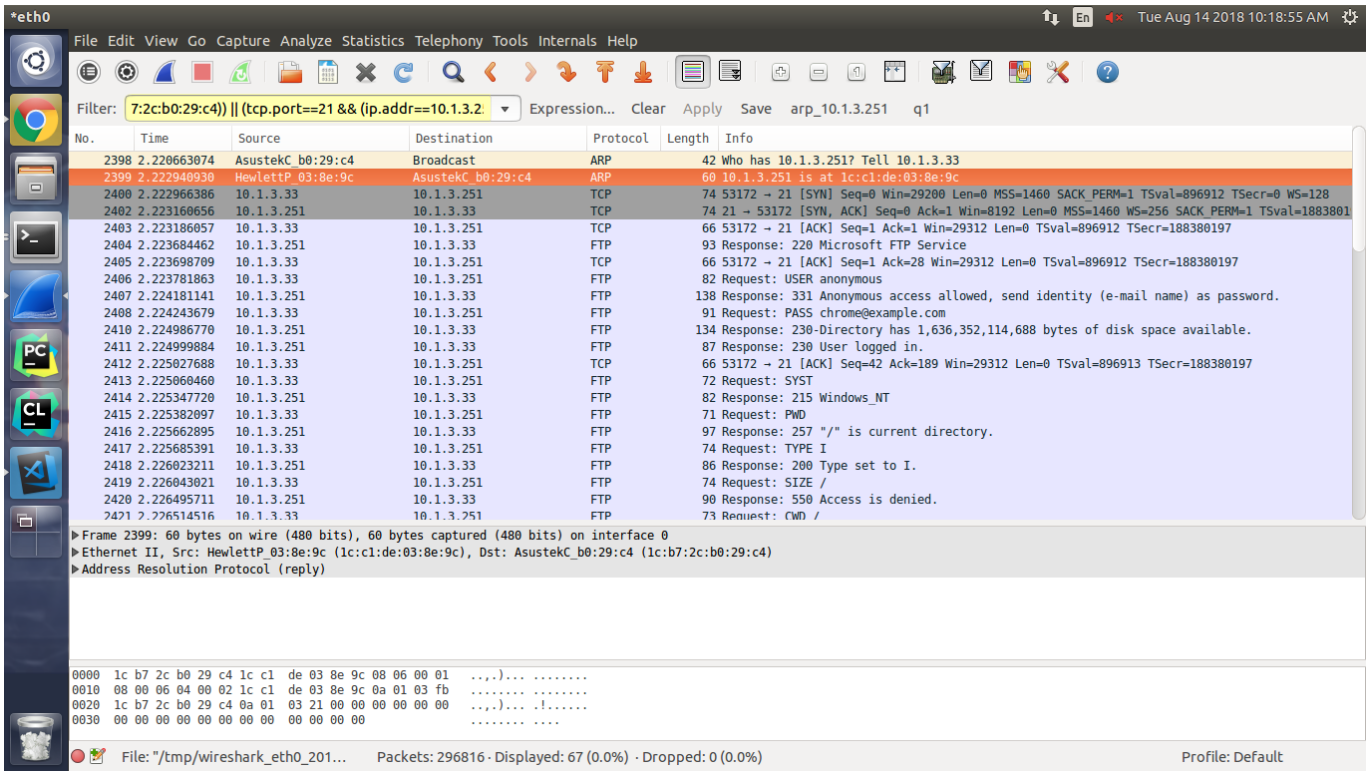
1. Perform request to local ip (ftp://10.1.3.251) and analyse `ARP` and `TCP` Protocol packets
2. Perform request to domain (ams.nirmauni.ac.in) at internal server and analyse `ARP-DNS-ARP-TCP-HTTP`
3. Perform request to any external server domain and analyse packets

Task 1: Request to 10.1.3.251

Filter Query - `ARP` - `TCP` - `TCP/HTTP`

```
    (arp.dst.proto_ipv4==10.1.3.251 && eth.src==1c:b7:2c:b0:29:c4 &&
eth.dst==ff:ff:ff:ff:ff:ff) || (arp.src.proto_ipv4==10.1.3.251 &&
eth.dst==1c:b7:2c:b0:29:c4) || (ip.addr==10.1.3.251 && tcp.port==21)
```

Result:



Explanation

Flow

First of all we require MAC Address (Ethernet card address) of target (destination) host to communicate at Data link layer hence we need to broadcast and as a result our system (operating system) will maintain ARP Table for target IP Address and MAC Address. As a response of ARP Query system will know the MAC Address of target host.

Filtering ARP query packets for 10.1.3.251

```
(arp.dst.proto_ipv4==10.1.3.251 && eth.src==1c:b7:2c:b0:29:c4 &&
```

```
eth.dst==ff:ff:ff:ff:ff:ff)
```

- with `arp.dst.proto_ipv4==10.1.3.251` we can filter wireshark packets to display only `ARP` request generated by our system (source machine) for the IPv4 Address `10.1.3.251`
- `eth.src==1c:b7:2c:b0:29:c4` MAC Address of our system (Source machine)
- `eth.dst==ff:ff:ff:ff:ff:ff` all bits set to high for broadcast

Filtering ARP response packets from 10.1.3.251

> Though the ARP query is filtered with above command; one needs to find out reponse of that too which will tell the MAC Address to source machine
>
> ```
> (arp.src.proto_ipv4==10.1.3.251 && eth.dst==1c:b7:2c:b0:29:c4)
> ```

- `arp.src.proto_ipv4==10.1.3.251` will filter arp packets with source (target machine IP) IPv4 address `10.1.3.251`
- `eth.dst==1c:b7:2c:b0:29:c4` will further add filter to display only ARP packets with destination MAC Address of source machine

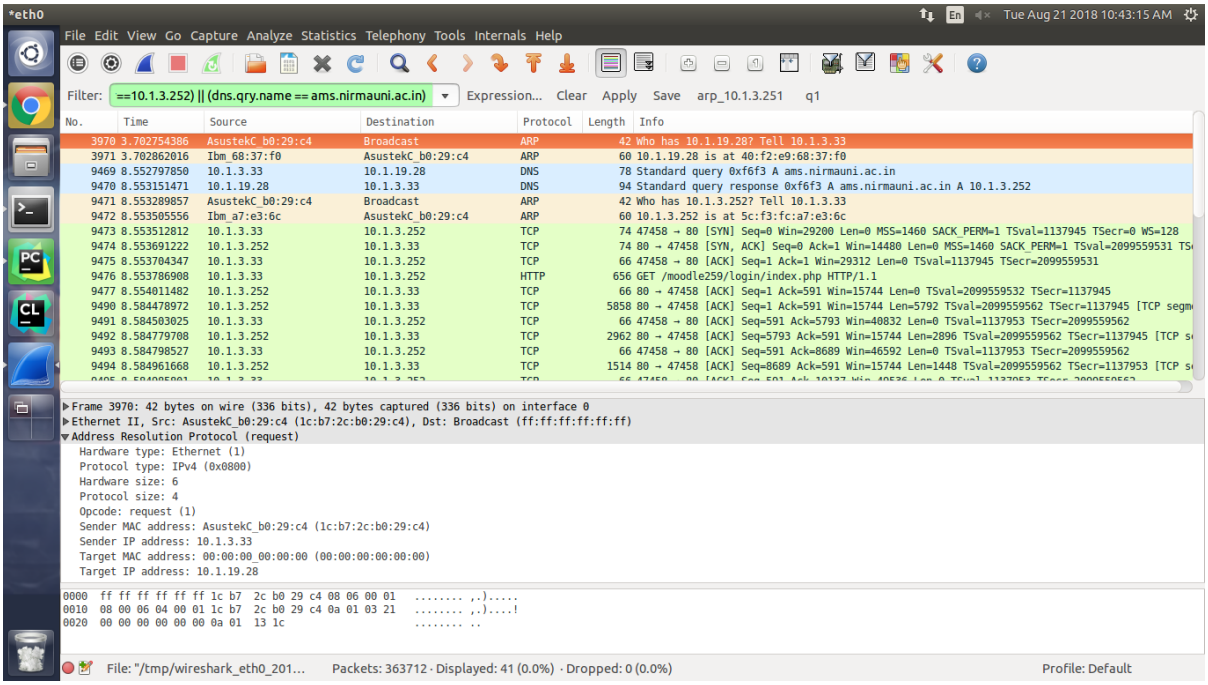
Task 2: Internal Server Request

request to TARGET : http://ams.nirmauni.ac.in/moodle259/

filter Query:

```
  ((arp.dst.proto_ipv4==10.1.19.28 && eth.src==1c:b7:2c:b0:29:c4 &&
eth.dst==ff:ff:ff:ff:ff:ff) || (arp.src.proto_ipv4==10.1.19.28 &&
eth.dst==1c:b7:2c:b0:29:c4) || (arp.dst.proto_ipv4==10.1.3.252 &&
eth.src==1c:b7:2c:b0:29:c4 && eth.dst==ff:ff:ff:ff:ff:ff) ||
(arp.src.proto_ipv4==10.1.3.252 && eth.src==1c:b7:2c:b0:29:c4))  ||
(arp.src.proto_ipv4==10.1.3.252 && eth.dst==1c:b7:2c:b0:29:c4) || ((http || tcp) &&
ip.addr==10.1.3.252) || (dns.qry.name == ams.nirmauni.ac.in)
```

Result:



Explanation:

Flow

- In order to achieve the connection with domain DNS comes in picture and hence source machine first need does `ARP` query broadcast for MAC Address of DNS Server `10.1.19.28`
- DNS will provide source machine the IPv4 Address(`10.1.3.252`) of target domain `ams.nirmauni.ac.in`
- Again system does ARP query for MAC Address of target IP `10.1.3.252` received from DNS

Action	Description
ARP to DNS	First source machine will generate arp query broadcast to get mac address of DNS Server `10.1.19.28`
DNS	After that source will communicate with DNS server with query of target domain and receive IP address for target domain
ARP to target IP	Now again source will resolve mac address of target with ARP protocol
TCP	source machine now starts to communicate with Target domain
HTTP	source is now communicating on HTTP protocol to send packets

ARP Query

```
(arp.dst.proto_ipv4==10.1.19.28 && eth.src==1c:b7:2c:b0:29:c4 &&
eth.dst==ff:ff:ff:ff:ff:ff) || (arp.src.proto_ipv4==10.1.19.28 &&
eth.dst==1c:b7:2c:b0:29:c4)
```

arp is filter keyword in wireshark to display only arp packets

Here the goal is to filter out packets which are resolving mac address of our destination

`eth.dst==ff:ff:ff:ff:ff:ff` will broadcast with all bits high from source machine

having mac address `1c:b7:2c:b0:29:c4` where `arp.dst.proto_ipv4` used to filter arp request for given destination

`arp.src.proto_ipv4=10.1.19.28` filters out arp request for IP Address `10.1.19.28`

`arp.src.proto_ipv4=10.1.3.252` filters out arp request for IP Address `10.1.3.252`

DNS Query

```
(dns.qry.name == ams.nirmauni.ac.in)
```

- `dns` is the filter keyword in wireshark to only display DNS entries
- above query will further filter out DNS packets to only those at which our target address: ams.nirmauni.ac.in is resolved; Hence we only get DNS result for our target domain

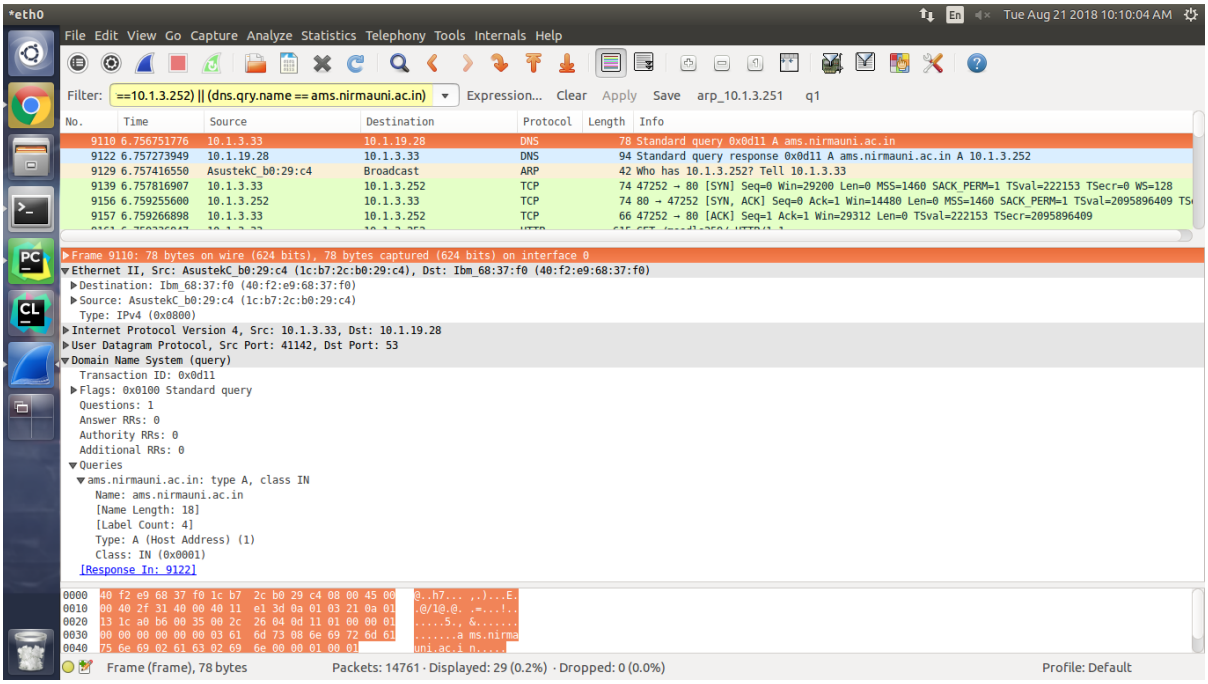


Fig: DNS query

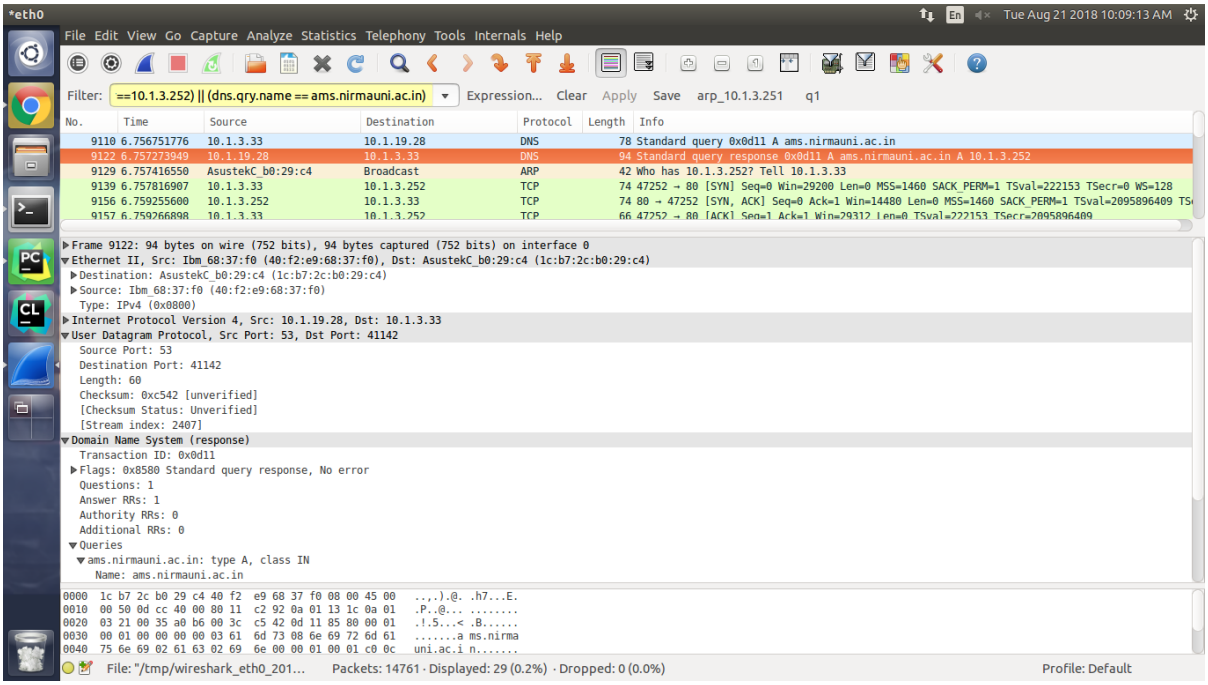


Fig: DNS response

TCP and HTTP connection

```
((http || tcp) && ip.addr==10.1.3.252)
```

above filter will filter all the incoming and outgoing http and tcp traffic for IP address 10.1.3.252 with the machine

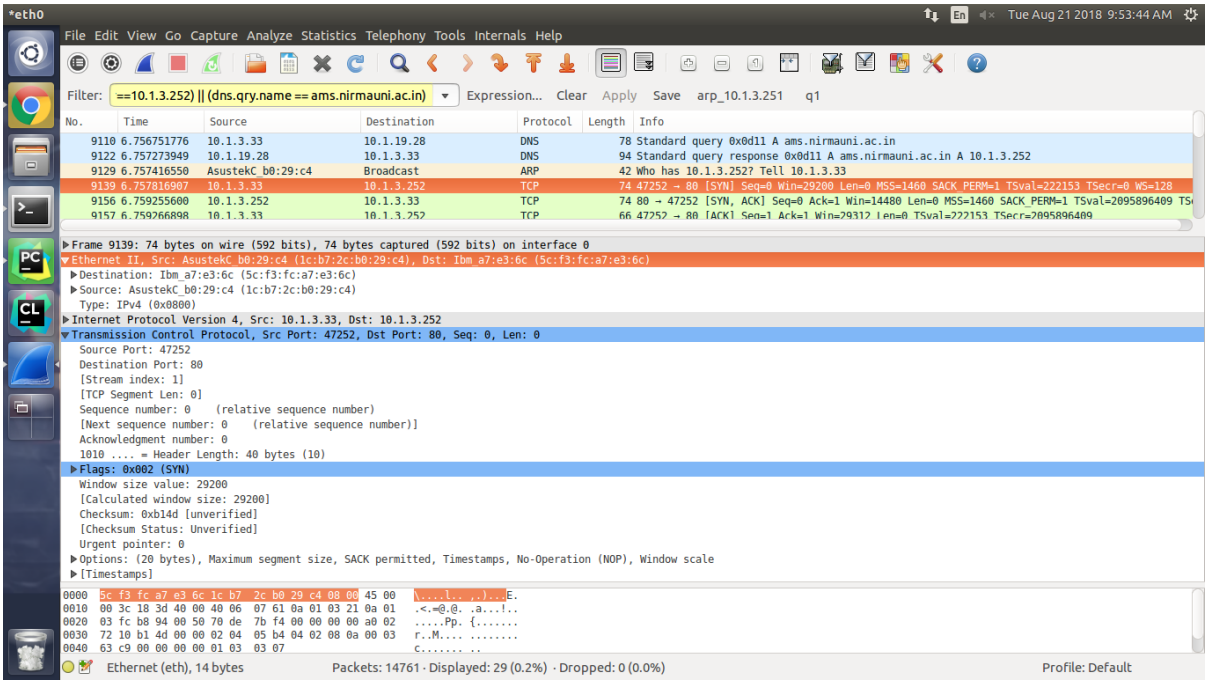


Fig: TCP SYNC

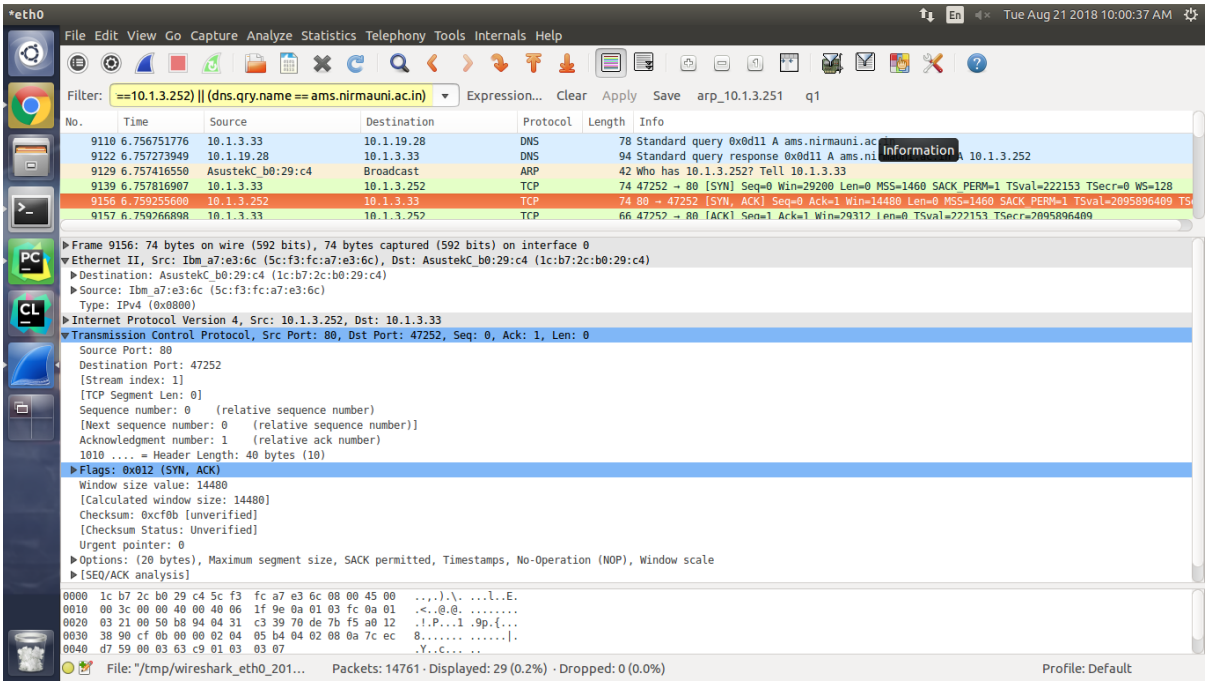


Fig: TCP SYNC + ACK

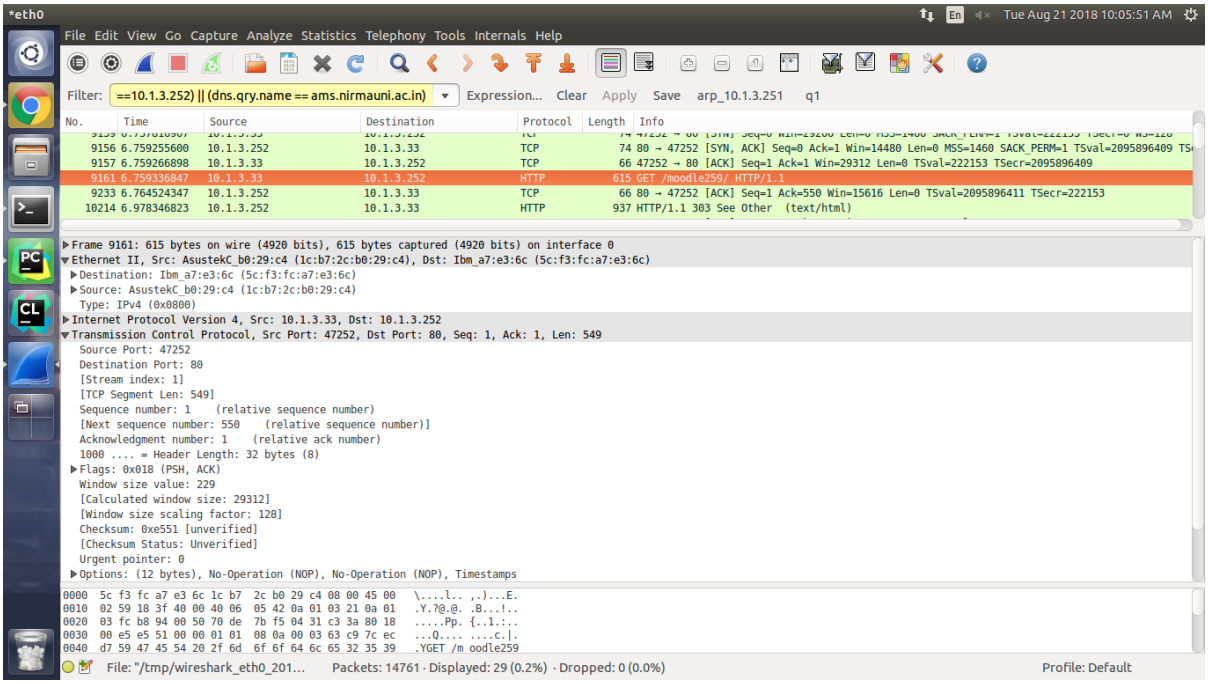


Fig: HTTP request

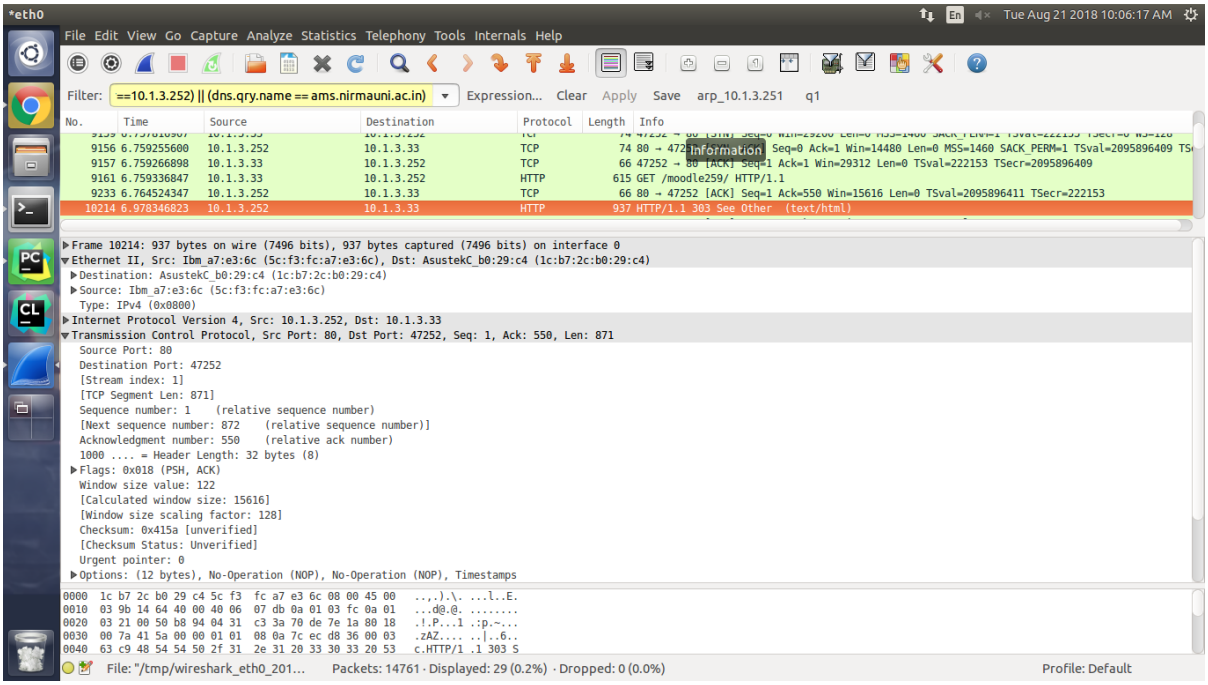


Fig: HTTP response

Task 3: External Server Request

request at TARGET: http://tony221b.pythonanywhere.com/

Before proceeding to this scenario first it is necessary to know a bit about established network condition.

- the target domain `tony221b.pythonanywhere.com` is outside the local network, source machine IPv4 address is `200.200.200.9` with default gateway `200.200.200.1` the DNS Server `8.8.8.8` is also not part of WAN network hence source machine can not communicate with it directly with MAC Address hence our system won't do any ARP query. Ultimately system will only do ARP query of gateway(router) and further on packets are transmitted by router and source machine is only communicating with IPv4 Address of target machine.

Filter Query

```
(arp.dst.proto_ipv4==200.200.200.1 && eth.dst==ff:ff:ff:ff:ff:ff &&
eth.src==XX:XX:XX:XX:XX:XX)|| (arp.src.proto_ipv4==200.200.200.1 &&
eth.dst==XX:XX:XX:XX:XX:XX) ||(dns.qry.name == tony221b.pythonanywhere.com) ||
(ip.addr==35.173.69.207)
```

Result:

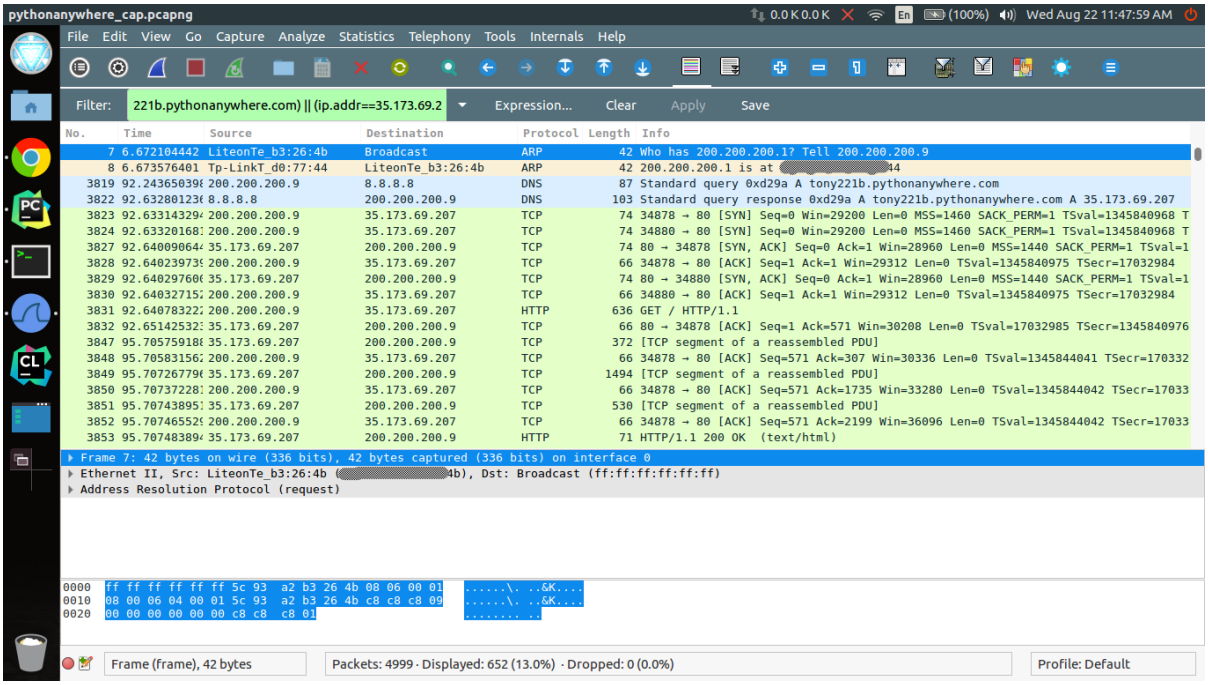


Fig: Captured packets for external server communication

Action	Description
ARP to Gateway	First source machine will generate ARP query broadcast to get mac address of gateway router `200.200.200.1`
DNS	After that source will communicate with DNS server `8.8.8.8` with query of target domain and receive IP address for target domain
TCP	source machine now starts to communicate with Target domain
HTTP	source is now communicating on HTTP protocol to send packets

ARP query to router

```
(arp.dst.proto_ipv4==200.200.200.1 && eth.dst==ff:ff:ff:ff:ff:ff &&
eth.src==XX:XX:XX:XX:XX:XX)|| (arp.src.proto_ipv4==200.200.200.1 &&
eth.dst==XX:XX:XX:XX:XX:XX)
```

- As explained previously above query will resolve address for default gateway (Router) as shown in [`Fig: Captured packets for external server communication`]

DNS Query

Now host machine needs IPv4 address of domain `tony221b.pythonanywhere.com` before establishing connection in this case where DNS Server `8.8.8.8` is outside local network no ARP result can be found for DNS Server source machine is communicating with DNS Server with IPv4 Address

```
dns.qry.name == tony221b.pythonanywhere.com
```

above filter will display dns query for target domain (tony221b.pythonanywhere.com)

Result:

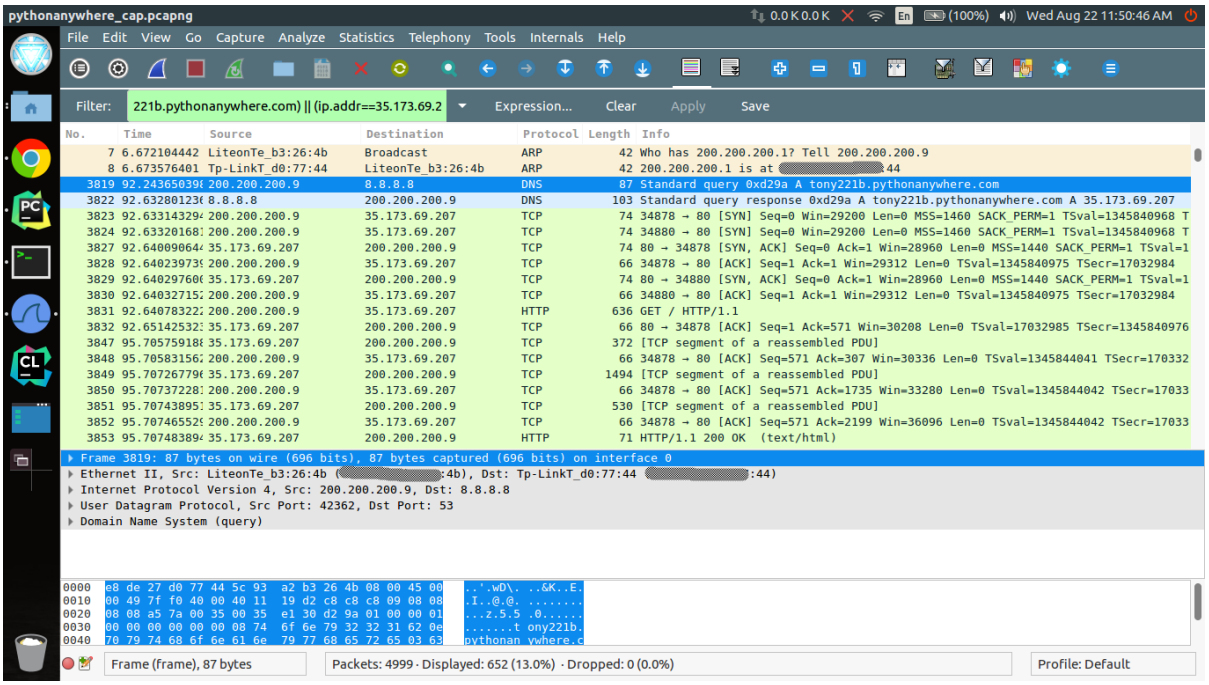


Fig: DNS Query to 8.8.8.8 from source machine

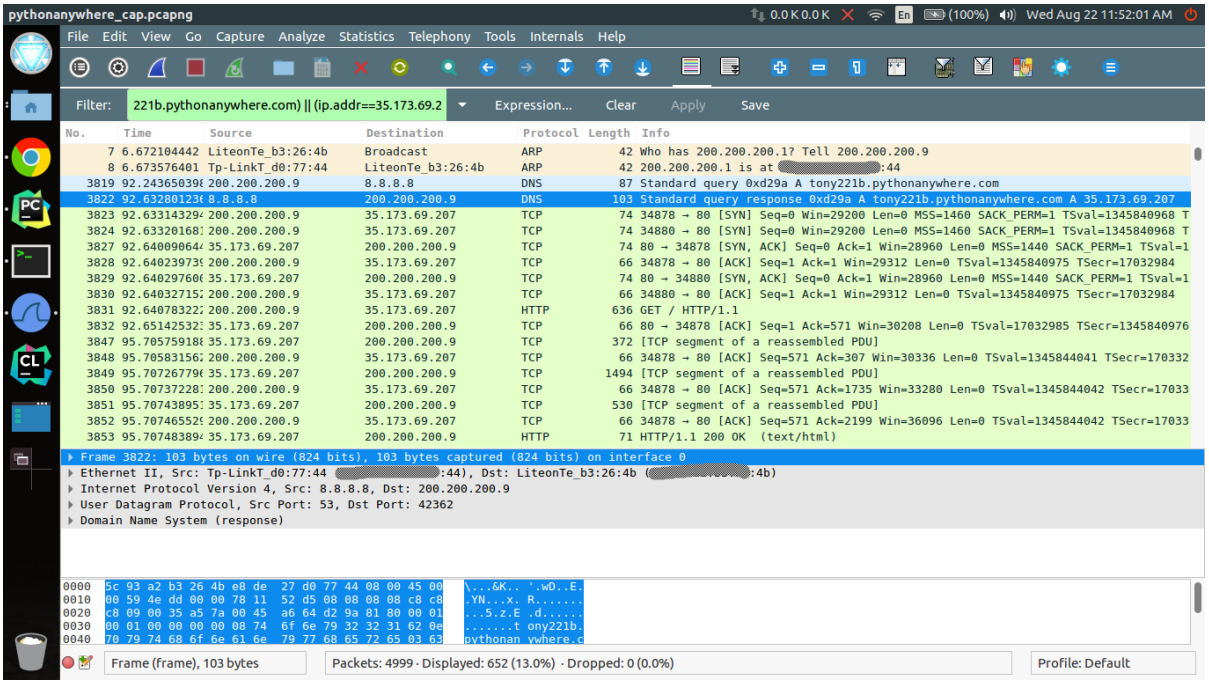


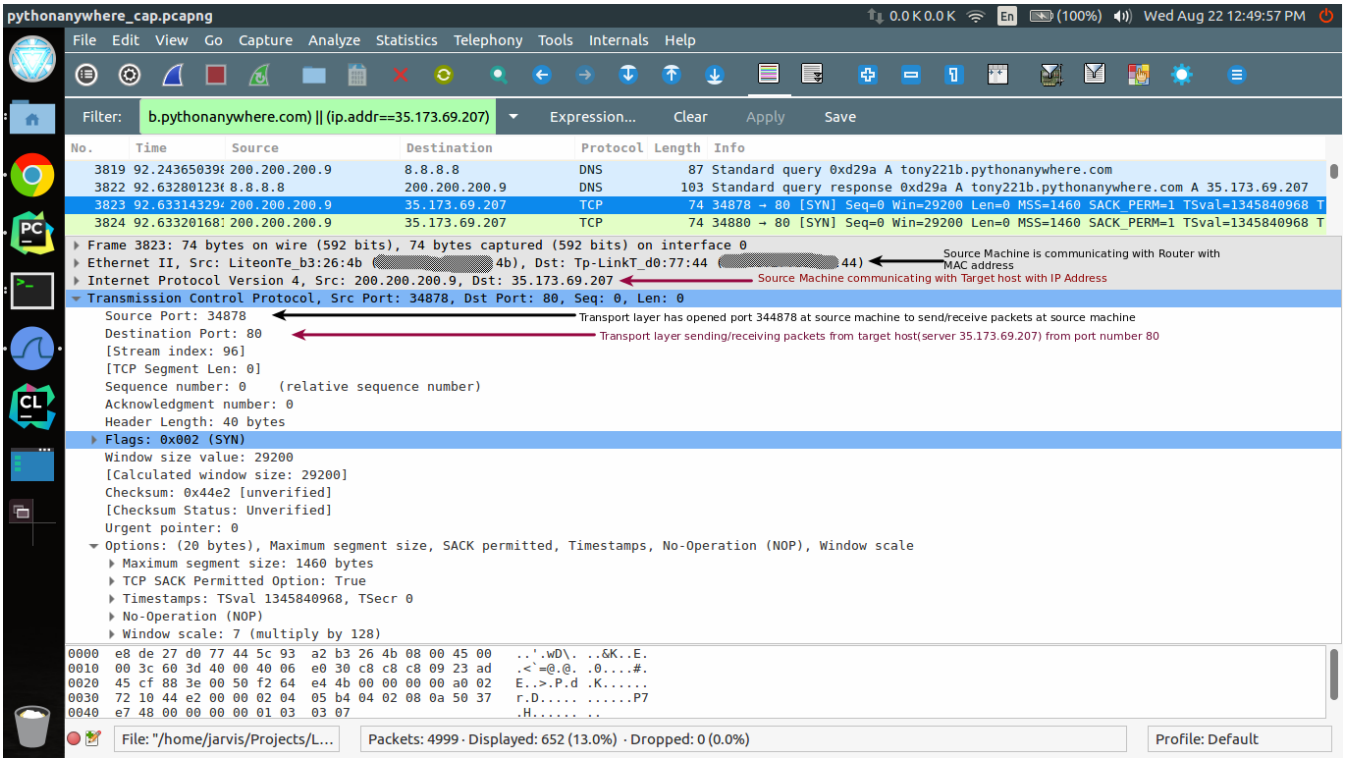
Fig: DNS Query response from 8.8.8.8 to source machine

TCP and HTTP connection

```
((http || tcp) && ip.addr==35.173.69.207)
```

- above filter will filter all the incoming and outgoing `http` and `tcp` traffic for IP address `35.173.69.207` with the machine

>



> Fig: TCP SYNC

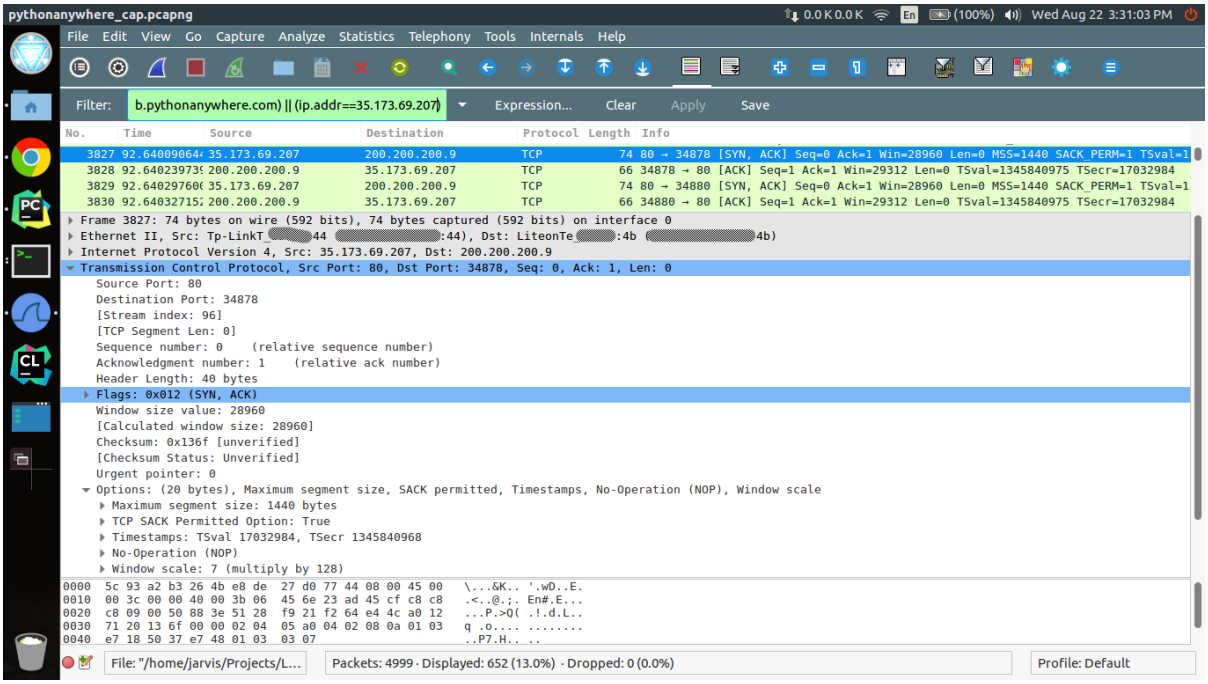


Fig: TCP SYNC + ACK

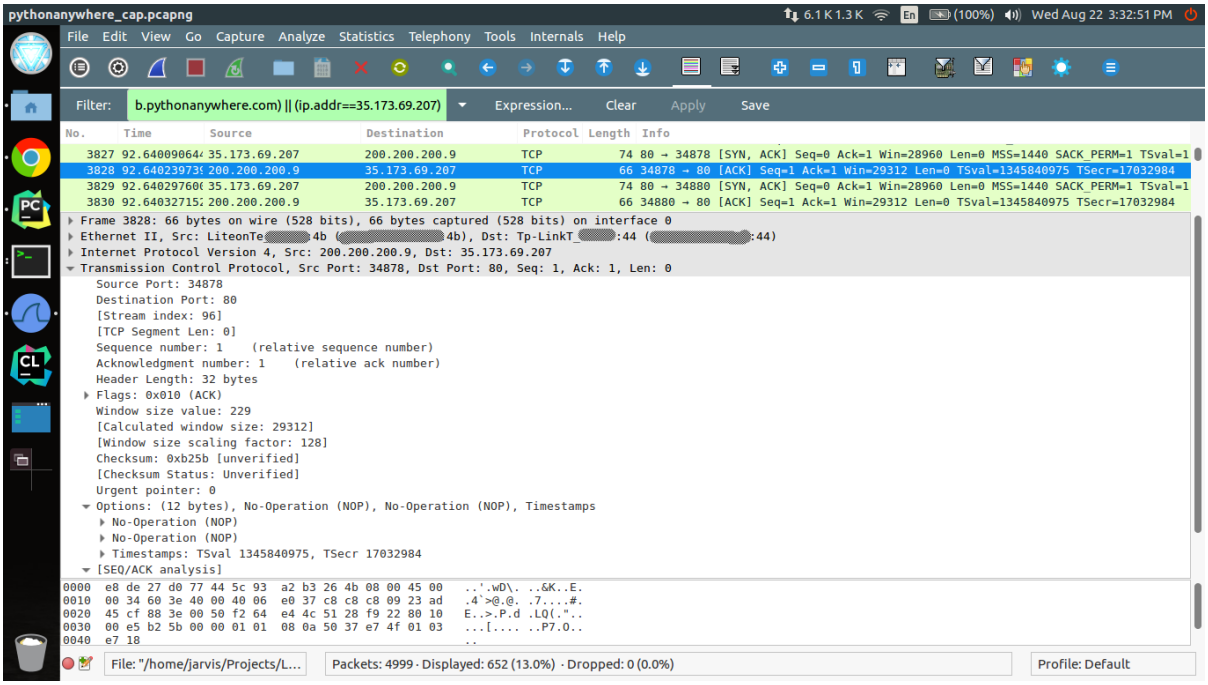


Fig: TCP ack

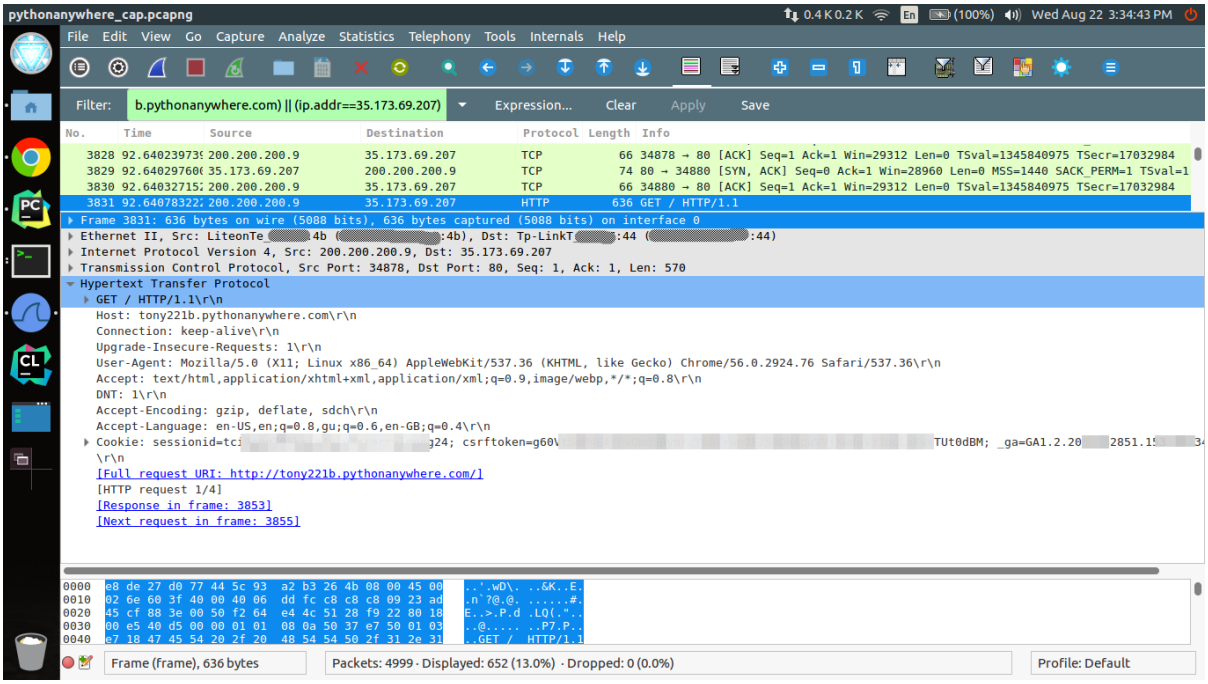


Fig: HTTP request

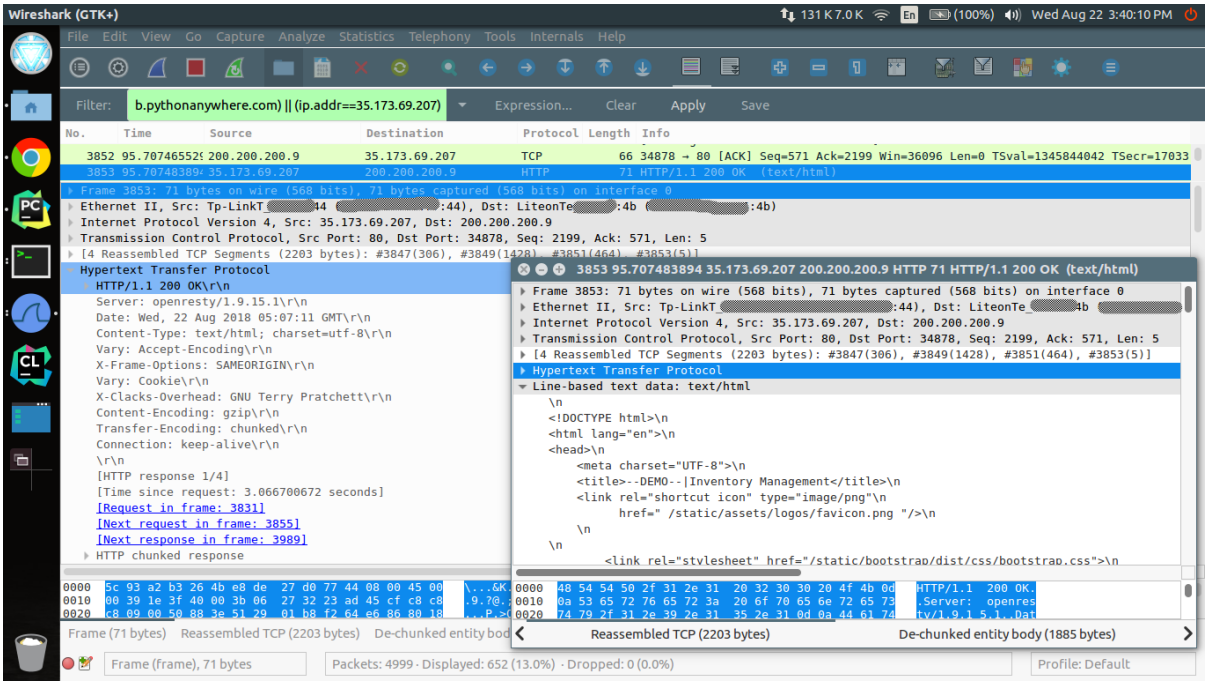


Fig: HTTP response