

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
```

Result:

The image shows a Wireshark network traffic capture. The filter bar at the top displays the filter: `7:2c:b0:29:c4 || (tcp.port==21 && (ip.addr==10.1.3.2))`. The packet list shows several packets, including an ARP request (No. 2398) and a series of FTP traffic packets (Nos. 2400-2471). The packet details pane for packet 2399 shows the Ethernet II frame, source MAC address 1c:b7:2c:b0:29:c4, and destination MAC address 01:00:00:00:00:00. The packet bytes pane shows the raw data of the ARP request.

No.	Time	Source	Destination	Protocol	Length	Info
2398	2.220663074	AsustekC_b0:29:c4	Broadcast	ARP	42	who has 10.1.3.251? Tell 10.1.3.33
2399	2.222940930	HewlettP_03:8e:9c	AsustekC_b0:29:c4	ARP	60	10.1.3.251 is at 1c:c1:de:03:8e:9c
2400	2.222966386	10.1.3.33	10.1.3.251	TCP	74	53172 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=896912 TSecr=0 WS=128
2402	2.223160656	10.1.3.251	10.1.3.33	TCP	74	21 → 53172 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=1883801
2403	2.223186057	10.1.3.33	10.1.3.251	TCP	66	53172 → 21 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=896912 TSecr=188380197
2404	2.223684462	10.1.3.251	10.1.3.33	FTP	93	Response: 220 Microsoft FTP Service
2405	2.223698709	10.1.3.33	10.1.3.251	TCP	66	53172 → 21 [ACK] Seq=1 Ack=28 Win=29312 Len=0 TSval=896912 TSecr=188380197
2406	2.223781863	10.1.3.33	10.1.3.251	FTP	82	Request: USER anonymous
2407	2.224181141	10.1.3.251	10.1.3.33	FTP	138	Response: 331 Anonymous access allowed, send identity (e-mail name) as password.
2408	2.224243679	10.1.3.33	10.1.3.251	FTP	91	Request: PASS chrome@example.com
2410	2.224998670	10.1.3.251	10.1.3.33	FTP	134	Response: 230 Directory has 1,636,352,114,688 bytes of disk space available.
2411	2.224999884	10.1.3.33	10.1.3.251	FTP	87	Response: 230 User logged in.
2412	2.225027688	10.1.3.33	10.1.3.251	TCP	66	53172 → 21 [ACK] Seq=42 Ack=189 Win=29312 Len=0 TSval=896913 TSecr=188380197
2413	2.225060460	10.1.3.33	10.1.3.251	FTP	72	Request: SYST
2414	2.225347720	10.1.3.251	10.1.3.33	FTP	82	Response: 215 Windows_NT
2415	2.225382097	10.1.3.33	10.1.3.251	FTP	71	Request: PWD
2416	2.225662895	10.1.3.251	10.1.3.33	FTP	97	Response: 257 "/" is current directory.
2417	2.225685391	10.1.3.33	10.1.3.251	FTP	74	Request: TYPE I
2418	2.226023211	10.1.3.251	10.1.3.33	FTP	86	Response: 200 Type set to I.
2419	2.226043021	10.1.3.33	10.1.3.251	FTP	74	Request: SIZE /
2420	2.226495711	10.1.3.251	10.1.3.33	FTP	90	Response: 550 Access is denied.
2421	2.226514516	10.1.3.33	10.1.3.251	FTP	73	Request: CWD /

Frame 2399: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: HewlettP_03:8e:9c (1c:c1:de:03:8e:9c), Dst: AsustekC_b0:29:c4 (1c:b7:2c:b0:29:c4)
Address Resolution Protocol (reply)

0000 1c b7 2c b0 29 c4 1c c1 de 03 8e 9c 00 00 00 01
0010 00 00 00 04 00 02 1c c1 de 03 8e 9c 0a 01 03 fb
0020 1c b7 2c b0 29 c4 0a 01 03 21 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

File: /tmp/wireshark_eth0_201... Packets: 296816 - Displayed: 67 (0.0%) - Dropped: 0 (0.0%) Profile: Default

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 response 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

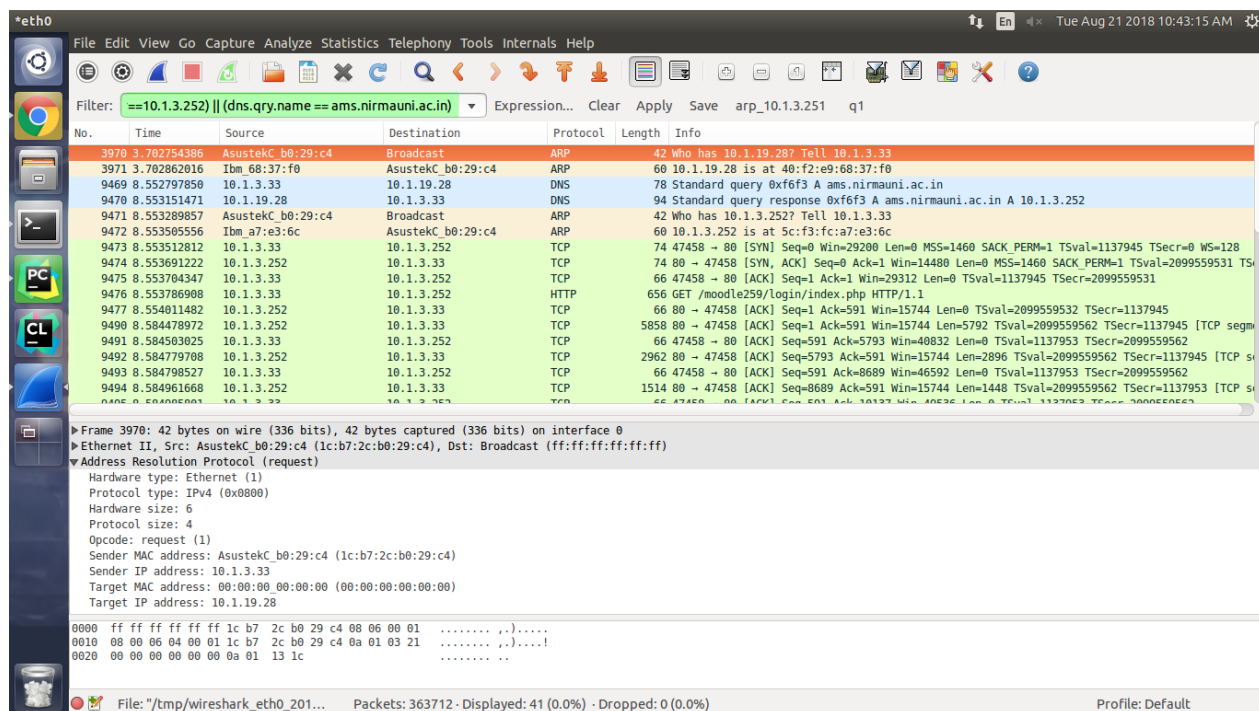
Task 2: Internal Server Request

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))
```

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

Action	Description
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 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

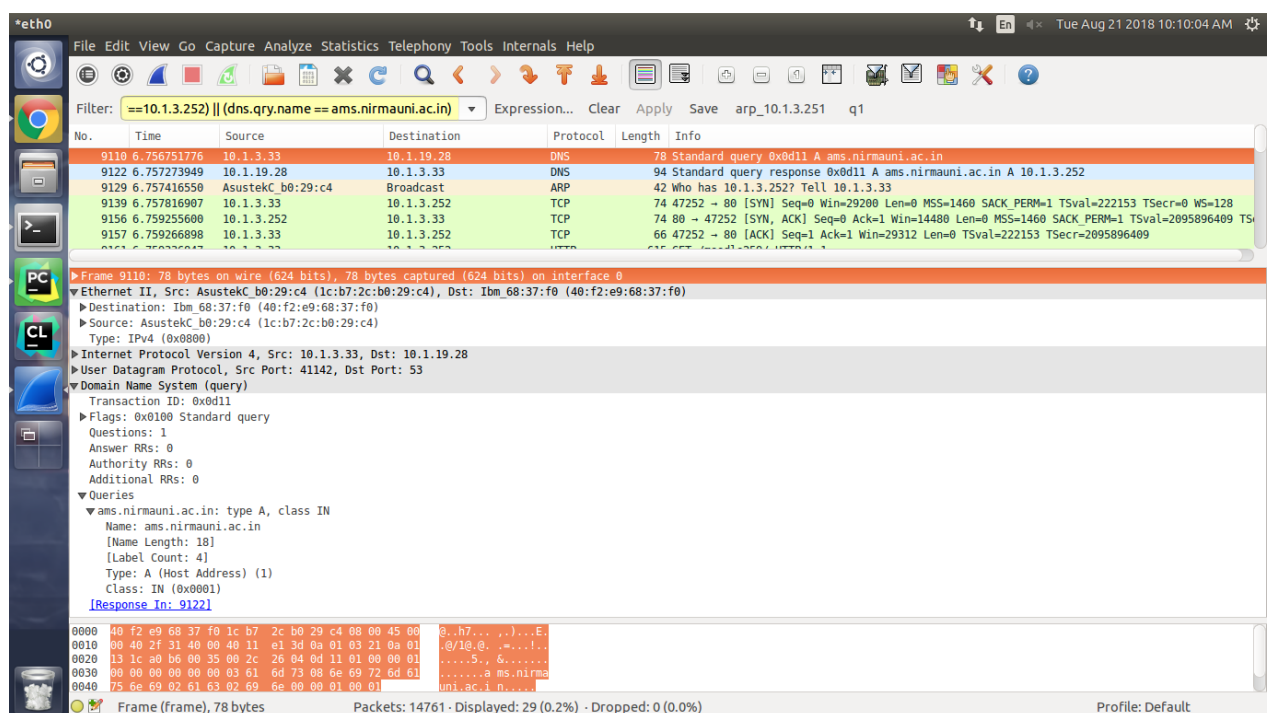


Fig: DNS query

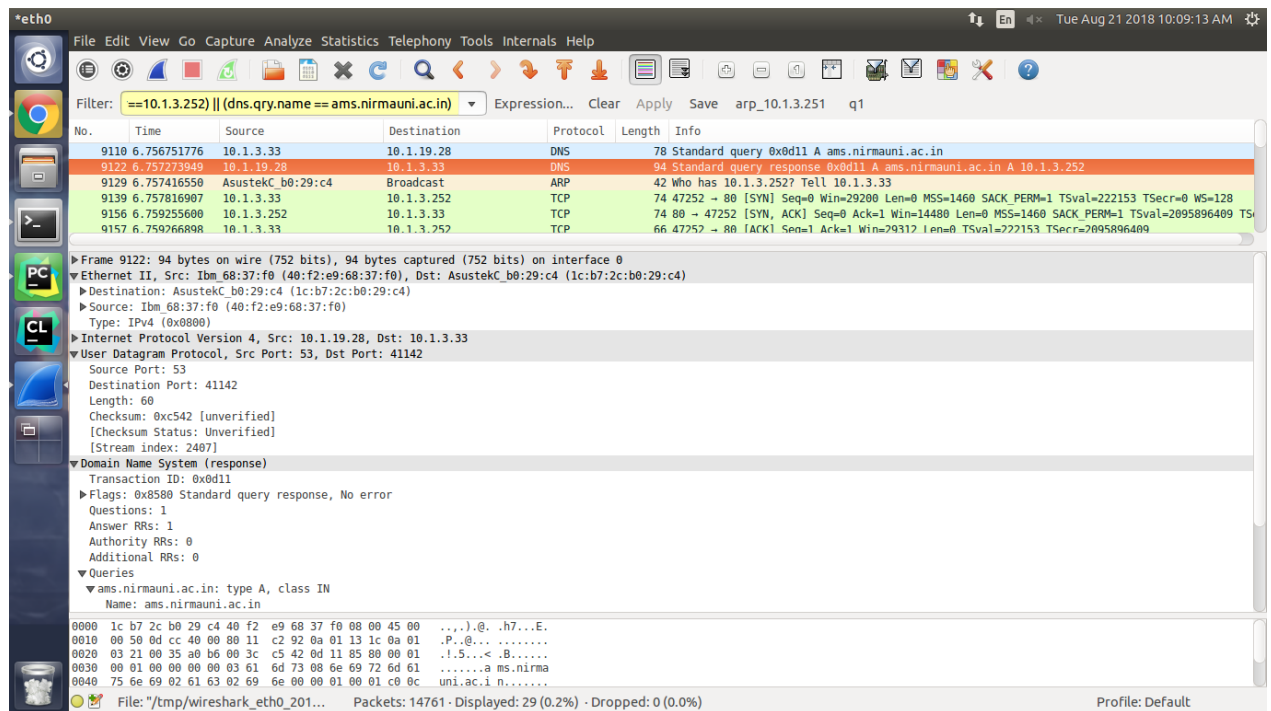


Fig: DNS response

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

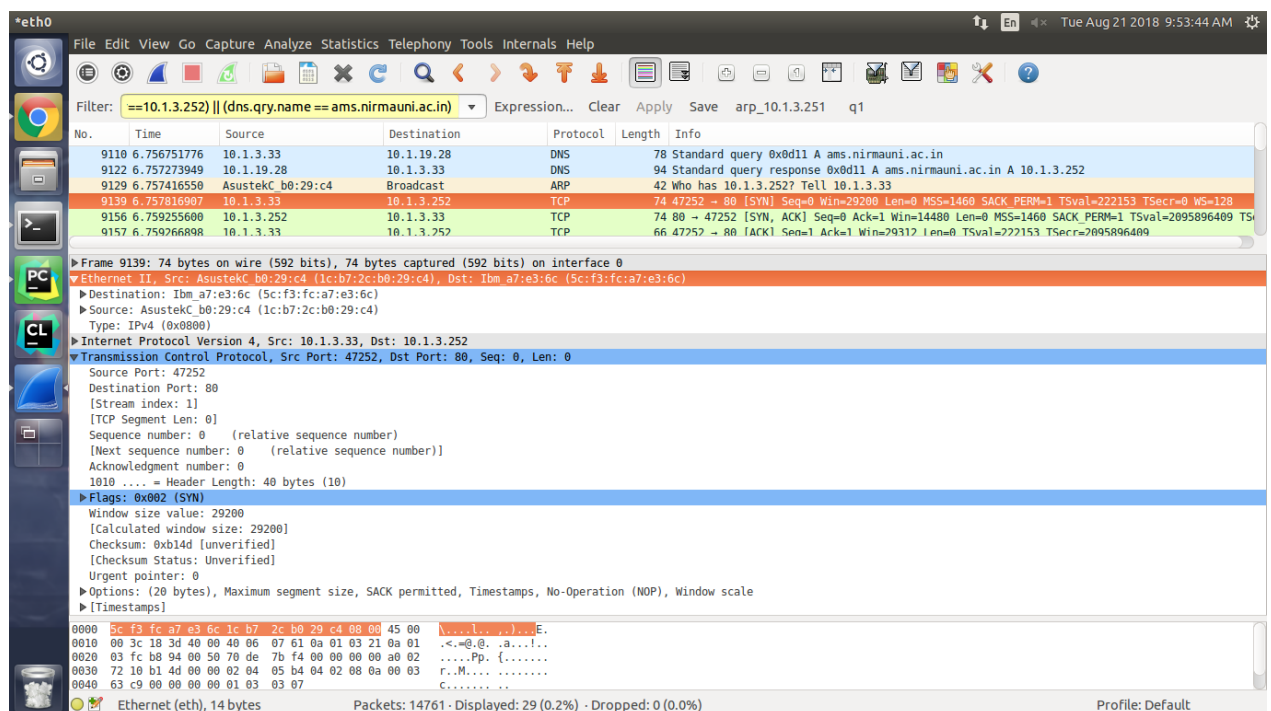


Fig: TCP SYNC

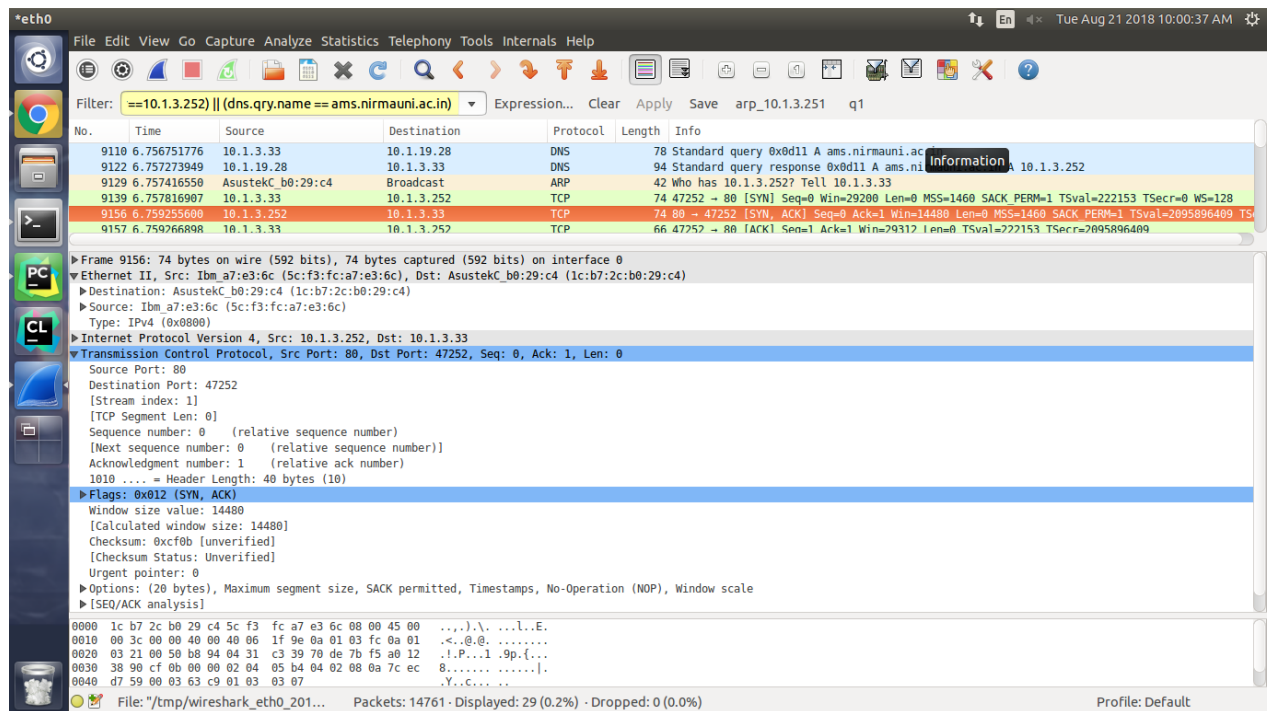


Fig: TCP SYNC + ACK

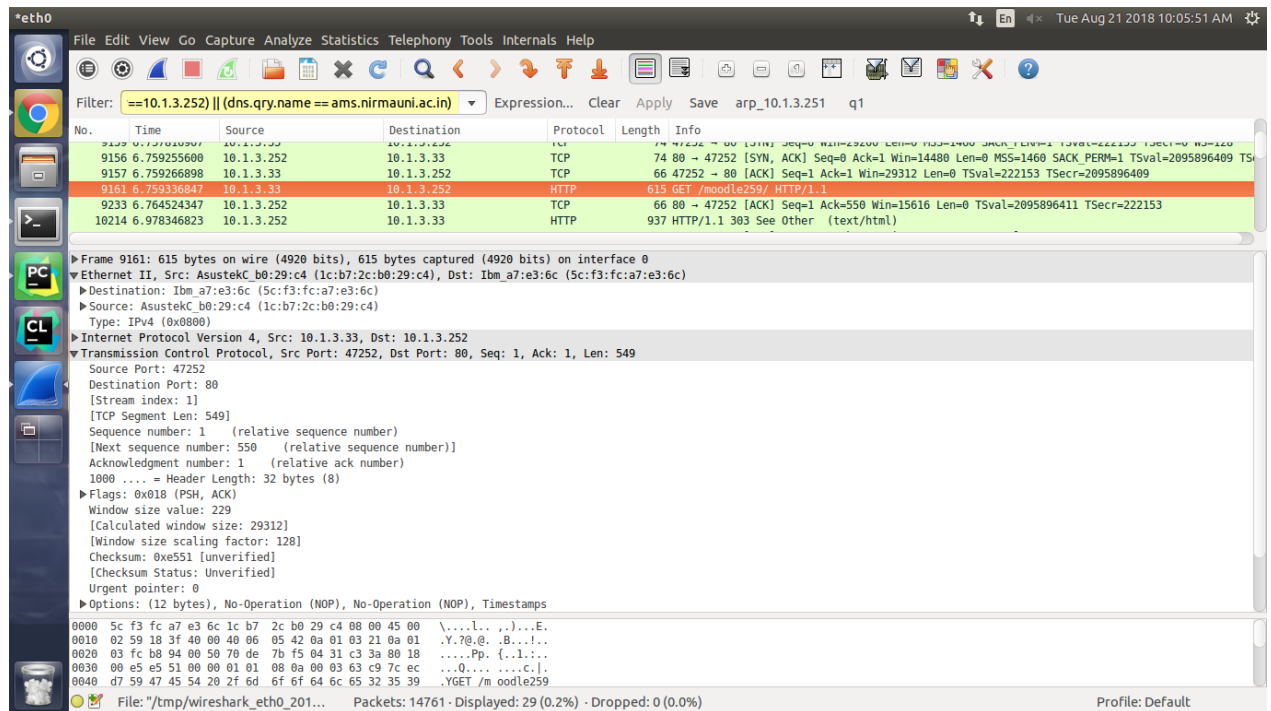


Fig: HTTP request

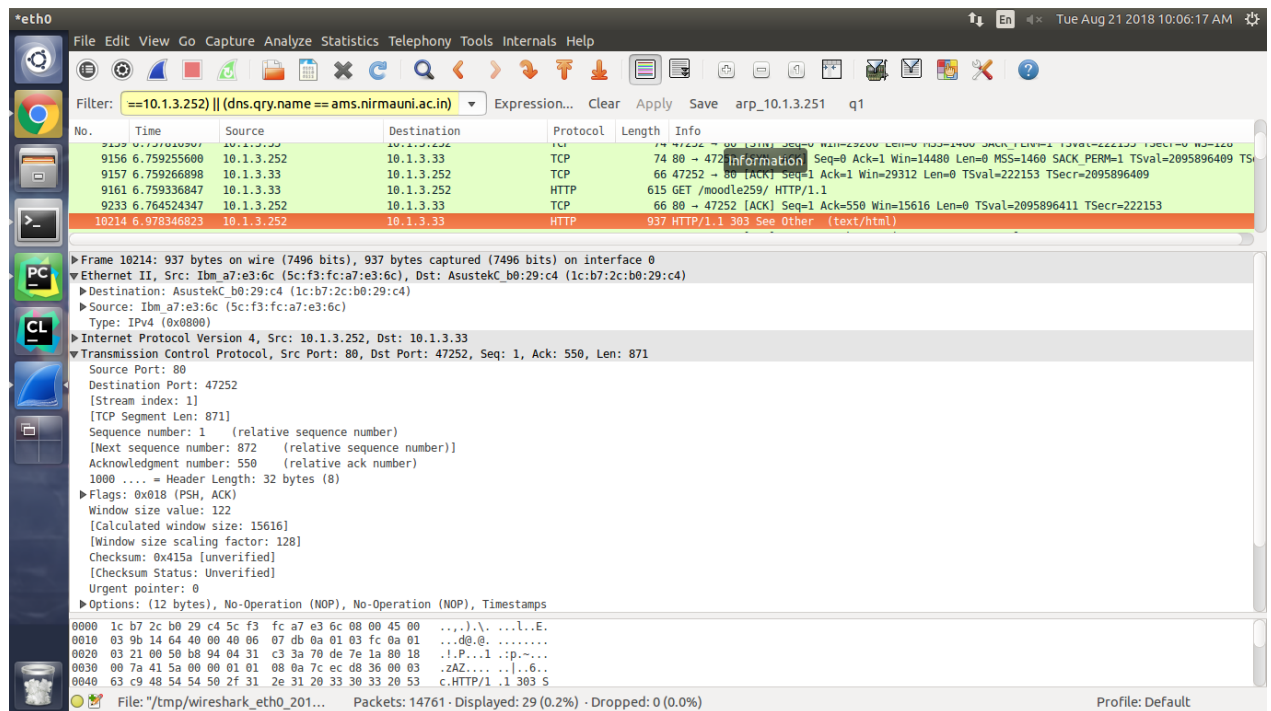


Fig: HTTP response

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:
```

Result:

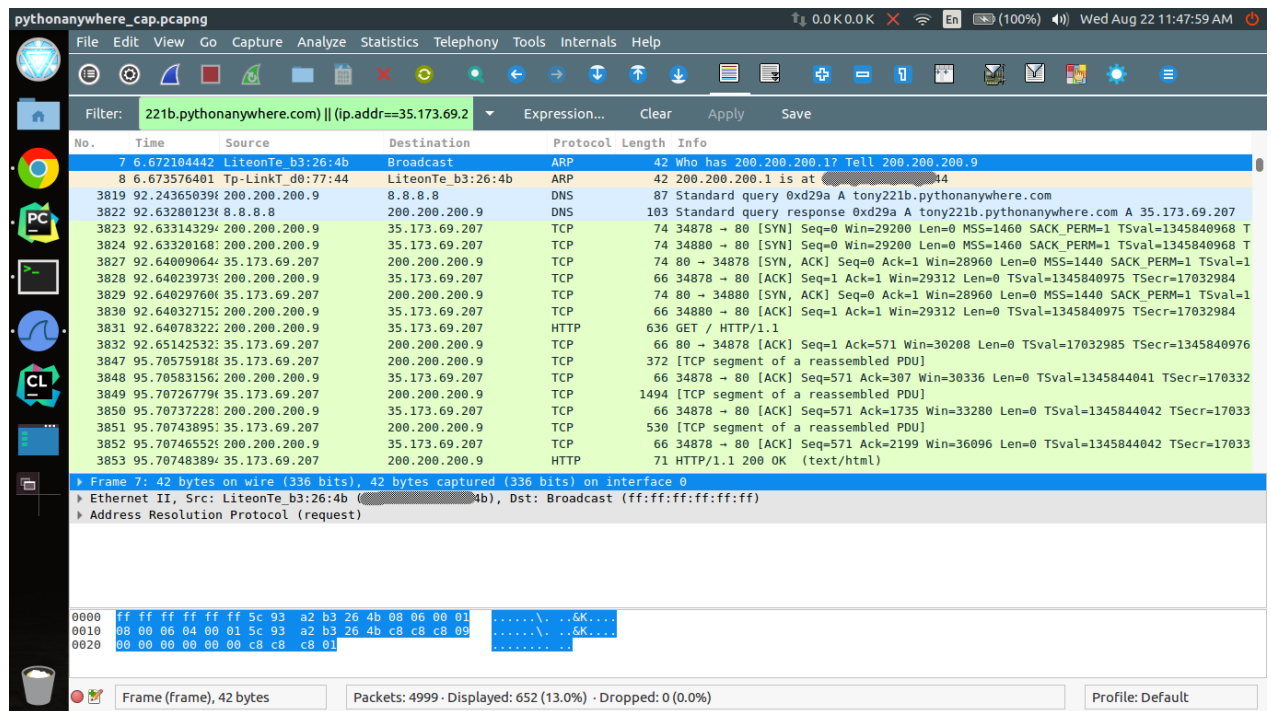


Fig: Captured packets for external server communication

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:
```

- 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:

The screenshot shows a Wireshark capture of network traffic. The filter is set to `221b.pythonanywhere.com || (ip.addr==35.173.69.2)`. The packet list shows a DNS query (No. 3819) from source 200.200.200.9 to destination 8.8.8.8. The packet details pane shows the query for `tony221b.pythonanywhere.com`. The packet bytes pane shows the raw data of the query.

No.	Time	Source	Destination	Protocol	Length	Info
7	6.672104442	LiteonTe_b3:26:4b	Broadcast	ARP	42	Who has 200.200.200.1? Tell 200.200.200.9
8	6.673576401	Tp-LinkT_d0:77:44	LiteonTe_b3:26:4b	ARP	42	200.200.200.1 is at [redacted]:44
3819	92.24365039f	200.200.200.9	8.8.8.8	DNS	87	Standard query 0xd29a A tony221b.pythonanywhere.com

Fig: DNS Query to 8.8.8.8 from source machine

The screenshot shows a Wireshark capture of network traffic. The filter is set to `221b.pythonanywhere.com || (ip.addr==35.173.69.2)`. The packet list shows a DNS response (No. 3822) from source 8.8.8.8 to destination 200.200.200.9. The packet details pane shows the response for `tony221b.pythonanywhere.com`. The packet bytes pane shows the raw data of the response.

No.	Time	Source	Destination	Protocol	Length	Info
7	6.672104442	LiteonTe_b3:26:4b	Broadcast	ARP	42	Who has 200.200.200.1? Tell 200.200.200.9
8	6.673576401	Tp-LinkT_d0:77:44	LiteonTe_b3:26:4b	ARP	42	200.200.200.1 is at [redacted]:44
3819	92.24365039f	200.200.200.9	8.8.8.8	DNS	87	Standard query 0xd29a A tony221b.pythonanywhere.com
3822	92.63280123f	8.8.8.8	200.200.200.9	DNS	103	Standard query response 0xd29a A tony221b.pythonanywhere.com A 35.173.69.207

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

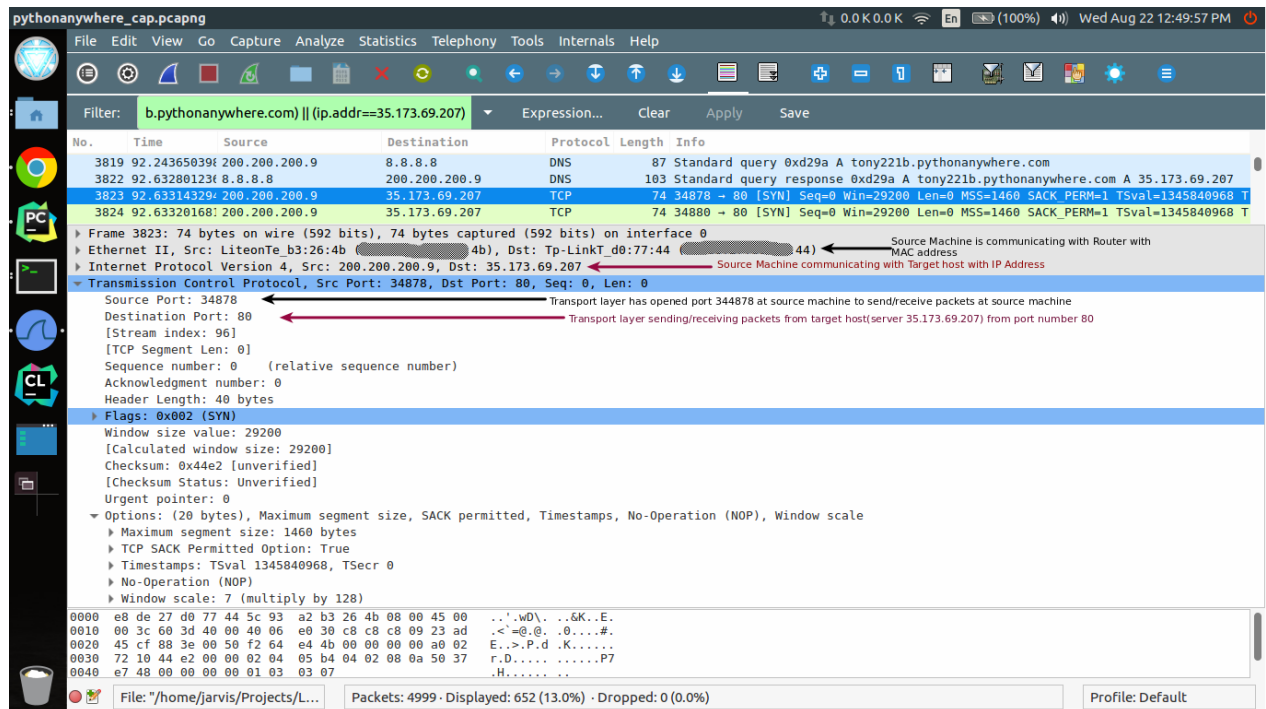


Fig: TCP SYNC

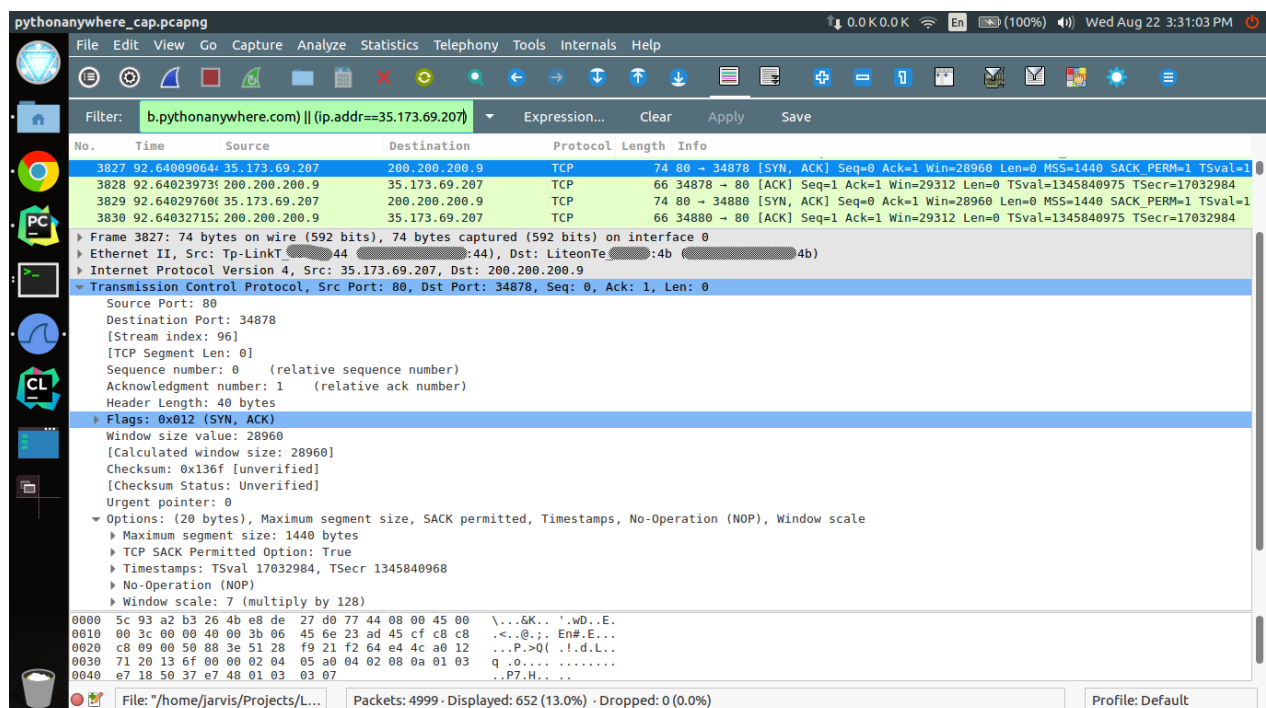


Fig: TCP SYNC + ACK

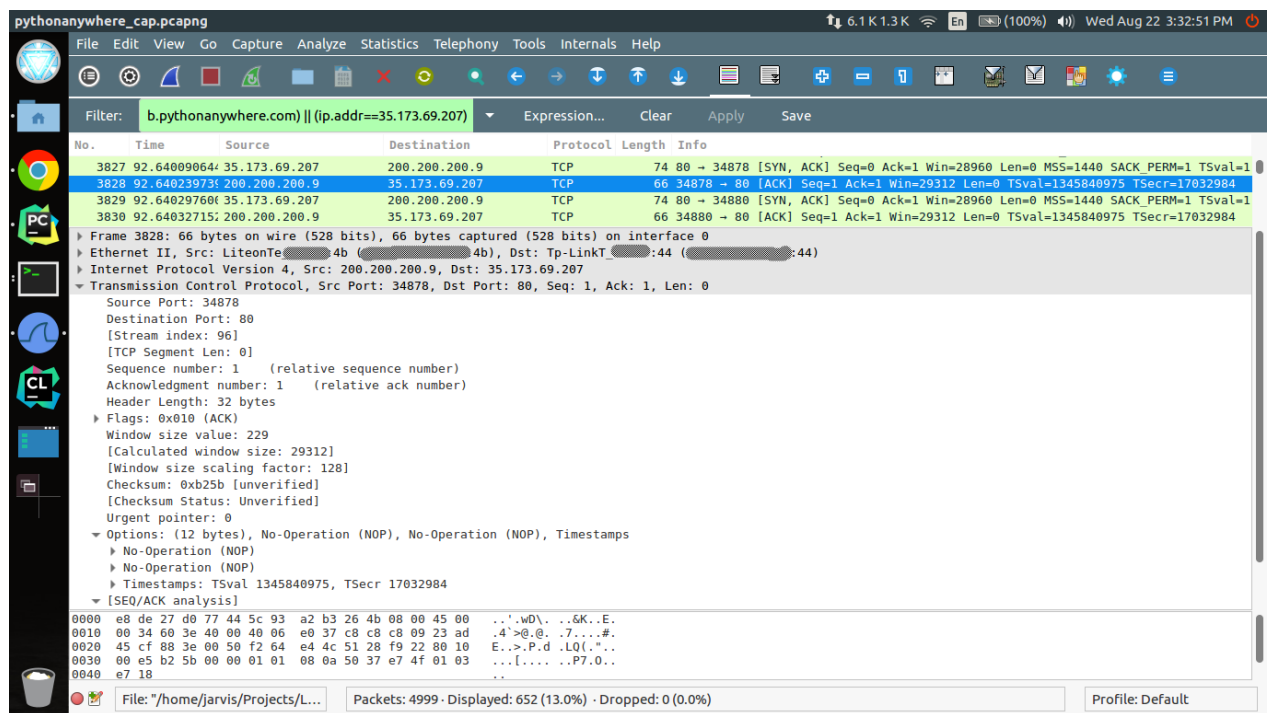


Fig: TCP ack

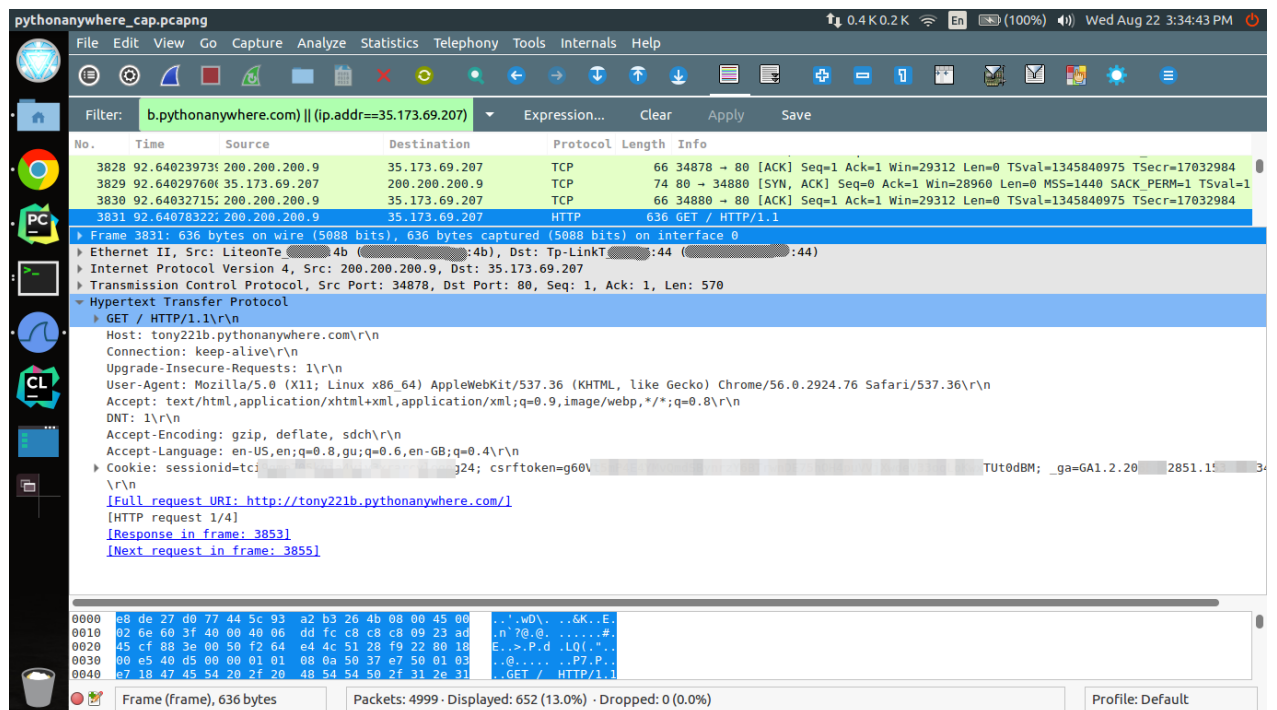


Fig: HTTP request

Wireshark (GTK+) interface showing a network capture. The filter is set to `b.pythonanywhere.com || (ip.addr==35.173.69.207)`. The capture shows a series of packets, with the selected packet being an HTTP response from 35.173.69.207 to 200.200.200.9.

The selected packet details are as follows:

- Frame 3853:** 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0
- Ethernet II, Src:** Tp-LinkT...:44, Dst: LiteonTe...:4b
- Internet Protocol Version 4, Src:** 35.173.69.207, Dst: 200.200.200.9
- Transmission Control Protocol, Src Port:** 80, Dst Port: 34878, Seq: 2199, Ack: 571, Len: 5
- [4 Reassembled TCP Segments (2203 bytes):** #3847(306), #3849(1428), #3851(464), #3853(5)]
- Hypertext Transfer Protocol**
 - HTTP/1.1 200 OK**
 - Server: openresty/1.9.15.1
 - Date: Wed, 22 Aug 2018 05:07:11 GMT
 - Content-Type: text/html; charset=utf-8
 - Vary: Accept-Encoding
 - X-Frame-Options: SAMEORIGIN
 - Vary: Cookie
 - X-Clacks-Overhead: GNU Terry Pratchett
 - Content-Encoding: gzip
 - Transfer-Encoding: chunked
 - Connection: keep-alive
 - [HTTP response 1/4]
 - [Time since request: 3.066700672 seconds]
 - [Request in frame: 3831]
 - [Next request in frame: 3855]
 - [Next response in frame: 3989]

The packet bytes pane shows the raw data of the response, including the HTML content:

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>--DEMO--Inventory Management</title>
<link rel="shortcut icon" type="image/png" href="/static/assets/logos/favicon.png"/>
<link rel="stylesheet" href="/static/bootstrap/dist/css/bootstrap.css">

```

The packet list pane shows the following information:

- Frame (71 bytes) Reassembled TCP (2203 bytes) De-chunked entity body (1885 bytes)
- Packets: 4999 - Displayed: 652 (13.0%) - Dropped: 0 (0.0%)
- Profile: Default

Fig: HTTP response