

PRACTICAL-12

AIM: Study use the wireshark for the various network protocols.

INTRODUCTION:

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible. One could think of a network packet analyzer as a measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable. Wireshark is available for free, is open source and is one of the best packet analyzer available today. Wireshark is available for windows, linux and mac operating system.

PACKET SNIFFER:

The basic tool for observing the messages exchanged between executing protocol entities is called a packet sniffer. As the name suggests, a packet sniffer captures ("sniffs") messages being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a copy of packets that are sent / received from/by application and protocols executing on your machine.

PURPOSES:

Here are some reasons people use Wireshark:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- QA engineers use it to verify network applications
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals.

FEATURES:

The following are some of the many features Wireshark provides:

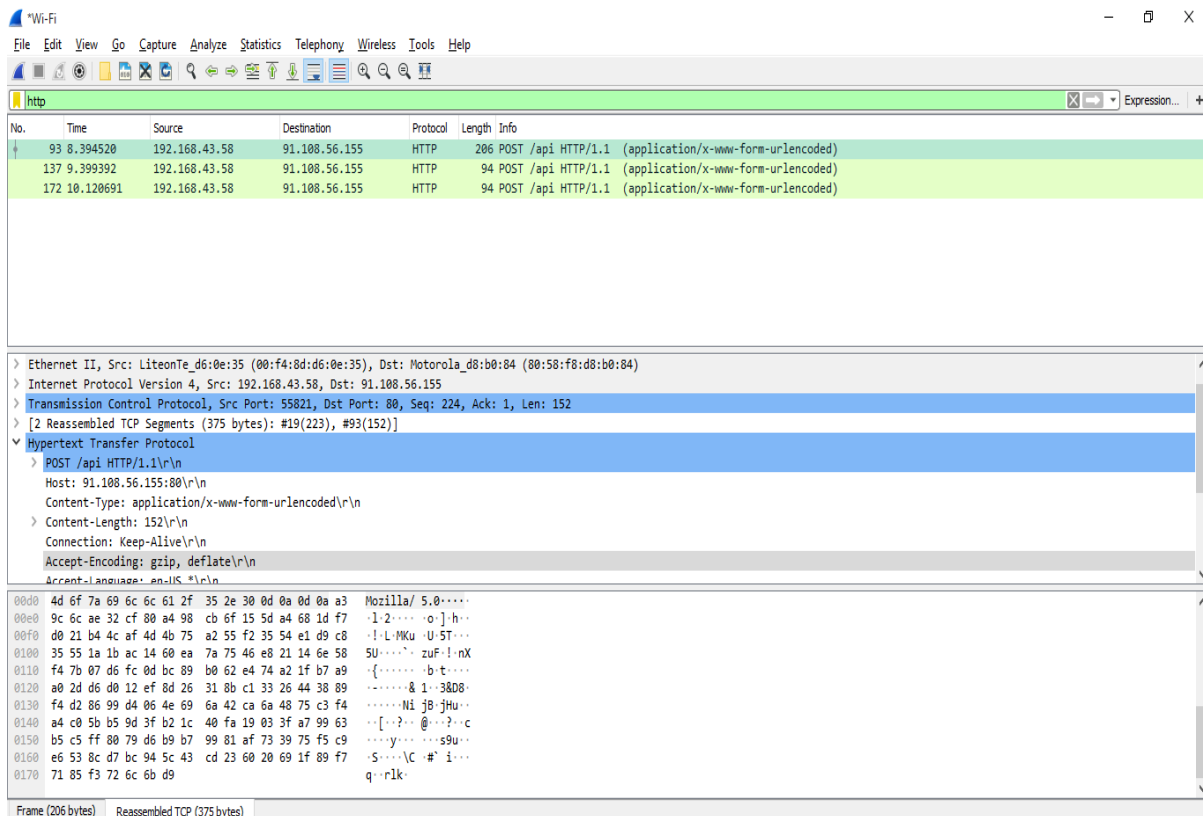
- Available for UNIX and Windows.
- Capture live packet data from a network interface.
- Open files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture programs.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
- Save packet data captured.
- Export some or all packets in a number of capture file formats.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.

- Create various statistics.
- ...and a lot more!

Wireshark can be downloaded from following link: <https://wireshark.org>

HTTP Protocol :

The **Hypertext Transfer Protocol (HTTP)** is an application layer protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlink to other resources that the user can easily access.



TCP Protocol :

TCP (Transmission Control Protocol) is connection-oriented, and a connection between client and server is established before data can be sent. The server must be listening (passive open) for connection requests from clients before a connection is established. Three-way handshake (active open), retransmission, and error-detection adds to reliability but lengthens latency. Applications that do not require reliable data stream service may use the User Datagram Protocol (UDP), which provides a connectionless datagram service that prioritizes time over reliability. TCP employs network congestion avoidance. However, there are vulnerabilities to TCP including denial of service, connection hijacking, TCP veto, and reset attack. For network security, monitoring, and debugging, TCP traffic can be intercepted and logged with a packet sniffer.

Wireshark packet capture showing a TCP connection. The packet list shows a sequence of packets including a SYN, ACK, and FIN. The packet details pane shows the structure of a TCP segment. The packet bytes pane shows the raw data in hexadecimal and ASCII.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------|---------------|---------------|----------|--------|--|
| 127 | 9.210750 | 192.168.43.58 | 91.108.56.155 | TCP | 54 | 56644 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0 |
| 128 | 9.210818 | 192.168.43.58 | 91.108.56.155 | TCP | 54 | 56645 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 |
| 129 | 9.211034 | 91.108.56.155 | 192.168.43.58 | TCP | 54 | [TCP Out-Of-Order] 80 → 55797 [FIN, ACK] Seq=1 Ack=2 Win=11052 Len=0 |
| 130 | 9.211086 | 192.168.43.58 | 91.108.56.155 | TCP | 54 | [TCP ZeroWindow] 55797 → 80 [ACK] Seq=2 Ack=2 Win=0 Len=0 |
| 131 | 9.211377 | 192.168.43.58 | 91.108.56.155 | TCP | 276 | 56645 → 80 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=222 [TCP segment of a reassembled PDU] |
| 132 | 9.214456 | 192.168.43.58 | 91.108.56.155 | SSL | 159 | Continuation Data |
| 133 | 9.270866 | 91.108.56.155 | 192.168.43.58 | TCP | 54 | 80 → 55797 [RST, ACK] Seq=2 Ack=2 Win=0 Len=0 |
| 136 | 9.397630 | 192.168.43.58 | 91.108.56.155 | TCP | 54 | 56644 → 443 [FIN, ACK] Seq=106 Ack=1 Win=65536 Len=0 |
| 137 | 9.399392 | 192.168.43.58 | 91.108.56.155 | HTTP | 94 | POST /api HTTP/1.1 (application/x-www-form-urlencoded) |
| 138 | 9.402995 | 192.168.43.58 | 91.108.56.155 | TCP | 66 | 56646 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |

Frame 75: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0
 Ethernet II, Src: LiteonTe_d6:0e:35 (00:f4:8d:d6:0e:35), Dst: Motorola_d8:b0:84 (00:58:f8:d8:b0:84)
 Internet Protocol Version 4, Src: 192.168.43.58, Dst: 151.101.154.114
 Transmission Control Protocol, Src Port: 55938, Dst Port: 443, Seq: 1, Ack: 1, Len: 1

0000 80 58 f8 d8 b0 84 00 f4 8d d6 0e 35 00 00 45 00 ·X·····5···
 0010 00 29 57 da 40 00 00 06 85 3a c0 a8 2b 3a 97 65 ·)W@·····+·:e
 0020 9a 72 da 82 01 bb b9 91 2c 23 dd 80 1d 68 50 10 ·r·····,#···hP
 0030 00 fe d4 3f 00 00 00 ·?·····

UDP Protocol :

UDP(User Datagram Protocol) uses a simple connectionless communication model with a minimum of protocol mechanisms. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection.

Wireshark packet capture showing a UDP connection. The packet list shows a sequence of packets including a SYN, ACK, and FIN. The packet details pane shows the structure of a UDP segment. The packet bytes pane shows the raw data in hexadecimal and ASCII.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------|----------------------|----------------------|----------|--------|----------------------|
| 82 | 8.036023 | 2404:6800:4007:807:: | 2405:205:c963:26b6:: | UDP | 1392 | 443 → 51769 Len=1330 |
| 83 | 8.062065 | 2405:205:c963:26b6:: | 2404:6800:4007:807:: | UDP | 99 | 51769 → 443 Len=37 |
| 84 | 8.064924 | 2404:6800:4007:807:: | 2405:205:c963:26b6:: | UDP | 1392 | 443 → 51769 Len=1330 |
| 85 | 8.090631 | 2405:205:c963:26b6:: | 2404:6800:4007:807:: | UDP | 99 | 51769 → 443 Len=37 |
| 86 | 8.171025 | 2404:6800:4007:807:: | 2405:205:c963:26b6:: | UDP | 1392 | 443 → 51769 Len=1330 |
| 87 | 8.196729 | 2405:205:c963:26b6:: | 2404:6800:4007:807:: | UDP | 99 | 51769 → 443 Len=37 |
| 88 | 8.257800 | 2404:6800:4007:807:: | 2405:205:c963:26b6:: | UDP | 1392 | 443 → 51769 Len=1330 |
| 89 | 8.283532 | 2405:205:c963:26b6:: | 2404:6800:4007:807:: | UDP | 99 | 51769 → 443 Len=37 |
| 90 | 8.360897 | 2404:6800:4007:807:: | 2405:205:c963:26b6:: | UDP | 1392 | 443 → 51769 Len=1330 |
| 91 | 8.386343 | 2405:205:c963:26b6:: | 2404:6800:4007:807:: | UDP | 99 | 51769 → 443 Len=37 |

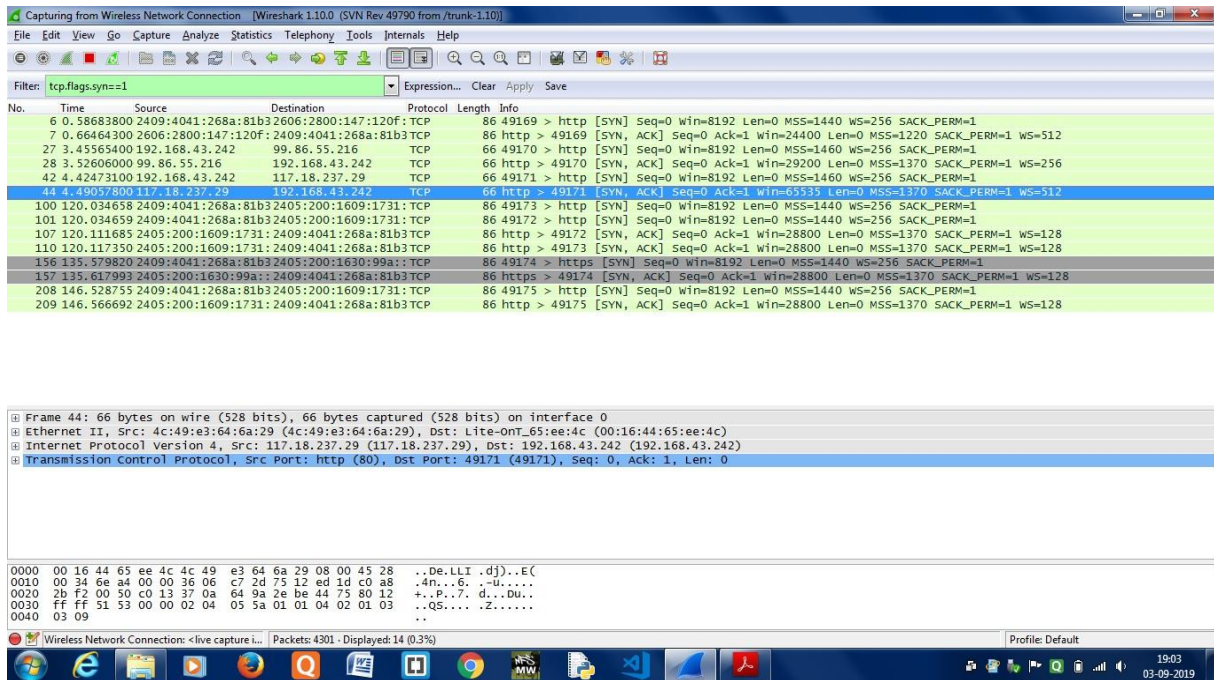
Frame 87: 99 bytes on wire (792 bits), 99 bytes captured (792 bits) on interface 0
 Ethernet II, Src: LiteonTe_d6:0e:35 (00:f4:8d:d6:0e:35), Dst: Motorola_d8:b0:84 (00:58:f8:d8:b0:84)
 Internet Protocol Version 6, Src: 2405:205:c963:26b6::, Dst: 2404:6800:4007:807::2004
 User Datagram Protocol, Src Port: 51769, Dst Port: 443
 Data (37 bytes)

0000 80 58 f8 d8 b0 84 00 f4 8d d6 0e 35 86 dd 60 00 ·X·····5···
 0010 00 00 00 2d 11 40 24 05 02 05 c9 63 26 b6 c1 be ····@5·····c&···
 0020 cc 93 40 3c a3 db 24 04 68 00 40 07 08 07 00 00 ·@<·\$·h@·····
 0030 00 00 00 20 04 ca 39 01 bb 00 2d 65 6f 5f 78 ····9·····ed·
 0040 f3 03 f4 0c 53 c2 18 73 0c 64 00 93 30 71 6a 64 ·f3·x·l·b·0·+·····
 0050 bd 7c 9d 8f 12 09 ee 2b ae d4 22 c5 ae 97 33 22 ·|·····+·····3·
 0060 c9 ca 52 ·-8

COMMANDS FOR WIRESHARK

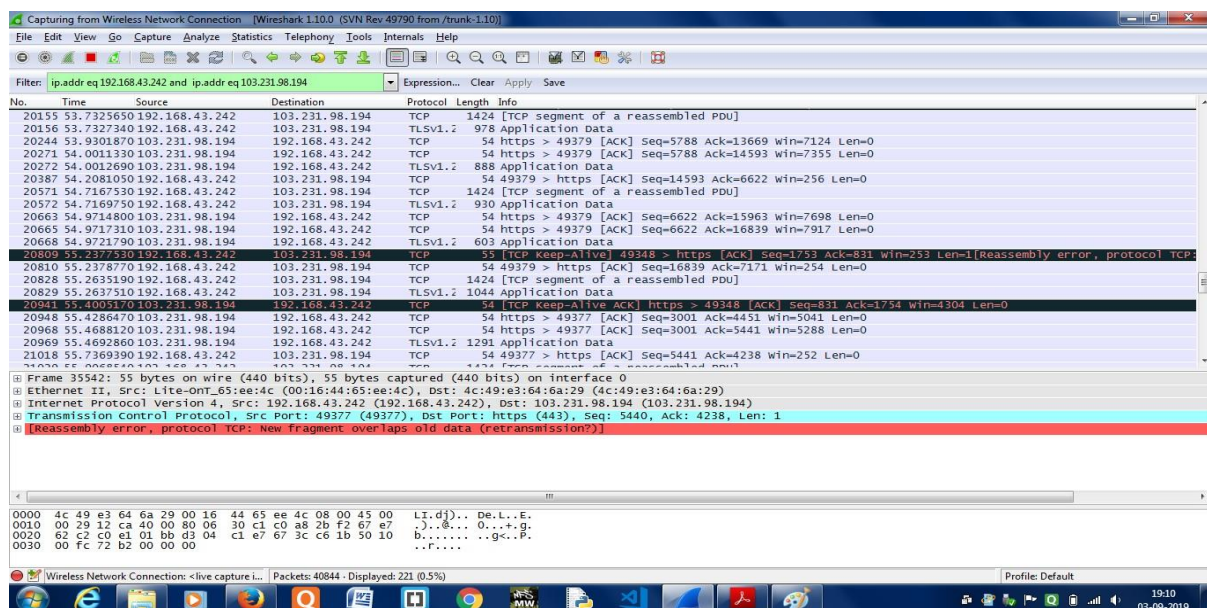
1) Analyzing TCP session using wireshark:

Find first SYN packet, sent from your PC to the web server. This signifies the start of a TCP 3-way handshake. Choose the correct flag and add==1.Hit the find button and first SYN packet in the trace should be highlighted.



2) Filtering particular IP address from given IP addresses:

This command helps a programmer to filter packets according to its requirement from available packets which are being travelling from source port to destination port. Thus, by using Wireshark we can analyse packets travelling from source to destination. `ip.addr eq 192.168.43.242 and ip.addr eq 103.231.98.194`



3)Analysing packets travelling from a particular source to particular destination:
The given command helps a programmer to trace those packets which travelling from given source ip address to given destination ip address.

ip.src==192.168.0.0/16 and ip.dst==192.168.0.0/16

| No. | Time | Source | Destination | Protocol | Length | Info |
|-------|------------|----------------|----------------|----------|--------|---|
| 3810 | 10.0212240 | 192.168.43.242 | 192.168.43.255 | NBNS | 92 | Name query NB HTTPS<00> |
| 4190 | 10.7778900 | 192.168.43.242 | 192.168.43.255 | NBNS | 92 | Name query NB HTTPS<00> |
| 4589 | 11.5421260 | 192.168.43.242 | 192.168.43.255 | NBNS | 92 | Name query NB HTTPS<00> |
| 7517 | 19.8265970 | 192.168.43.242 | 192.168.43.255 | NBNS | 92 | Name query NB HTTPS<00> |
| 7757 | 20.5903500 | 192.168.43.242 | 192.168.43.255 | NBNS | 92 | Name query NB HTTPS<00> |
| 8035 | 21.3546350 | 192.168.43.242 | 192.168.43.255 | NBNS | 92 | Name query NB HTTPS<00> |
| 64490 | 235.257468 | 192.168.43.1 | 192.168.43.242 | DHCP | 342 | DHCP ACK - Transaction ID 0xb6ffa53b |
| 64975 | 312.692696 | 192.168.43.242 | 192.168.43.255 | NBNS | 92 | Name query NB HTTPS<00> |
| 64995 | 313.443486 | 192.168.43.242 | 192.168.43.255 | NBNS | 92 | Name query NB HTTPS<00> |
| 64998 | 314.207701 | 192.168.43.242 | 192.168.43.255 | NBNS | 92 | Name query NB HTTPS<00> |
| 66572 | 509.283717 | 192.168.43.242 | 192.168.43.1 | DNS | 84 | Standard query 0xd938 A download.cdn.mozilla.net |
| 66654 | 509.965473 | 192.168.43.1 | 192.168.43.242 | DNS | 215 | Standard query response 0xd938 CNAME 2-01-2967-001e.cdx.cedexis.net CNAME cdn-mozilla-net.edgekey.net |
| 66655 | 509.965528 | 192.168.43.242 | 192.168.43.1 | ICMP | 243 | Destination unreachable (Port unreachable) |
| 67326 | 621.592026 | 192.168.43.1 | 192.168.43.242 | DHCP | 342 | DHCP ACK - Transaction ID 0x689a4100 |

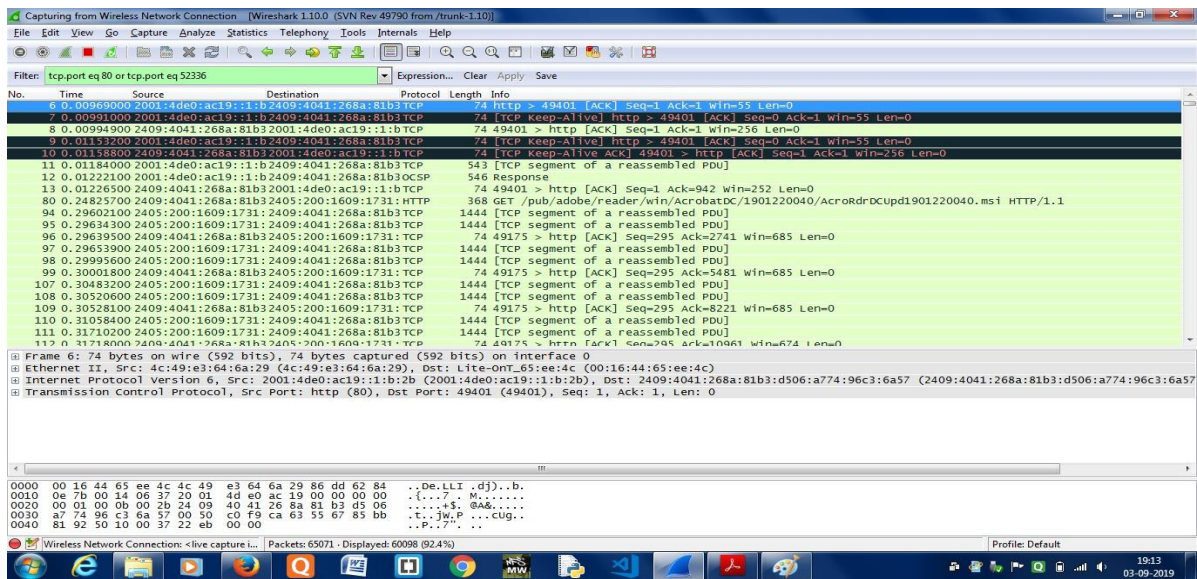
Frame 3810: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
 Ethernet II, Src: Lite-ONT_65:ee:4c (00:16:44:65:ee:4c), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 Internet Protocol Version 4, Src: 192.168.43.242 (192.168.43.242), Dst: 192.168.43.255 (192.168.43.255)
 User Datagram Protocol, Src Port: netbios-ns (137), Dst Port: netbios-ns (137)
 NetBIOS Name Service

| Offset | Hex | ASCII |
|--------|---|------------------------|
| 0000 | ff ff ff ff ff 00 16 44 65 ee 4c 08 00 45 00 |De.L..E. |
| 0010 | 00 4e 0e b9 00 00 80 11 52 a4 c0 a8 2b f2 c0 a8 | .N.....R...+... |
| 0020 | 2b ff 00 89 00 89 00 3a fc 0e e0 a4 01 10 00 01 | +.....E IFEFEFAF |
| 0030 | 00 00 00 00 00 00 20 45 49 46 45 46 45 46 41 46 |DCACACAC ACACACAC |
| 0040 | 44 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43 |DCACACAC ACACACAC |
| 0050 | 41 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43 |DCACACAC ACACACAC |

4)Monitoring traffic on more than two ports:

Sometimes you will be interested in inspecting traffic that matches either (or both) conditions whatsoever. For example if user wants to monitor tcp ports and other ports simultaneously then or relation can be used between them.

Tcp.port eq 80 or tcp.port eq 52336



CONCLUSION:

We have concluded that wireshark is a program that is used to capture data packets to allow a more precise analysis. Such a tool allows the user to examine his/her own computer for protocol errors and problems within the network architecture.