tcpdump - A Detailed Guide for Penetration Testing

Introduction

tcpdump is a powerful and widely-used network packet analyzer tool. It allows you to capture and analyze packets transmitted over a network in real-time. During penetration testing, tcpdump is a valuable tool for understanding the behavior of network traffic, detecting potential vulnerabilities, and gathering information about the network's structure, services, and security posture.

What is tcpdump?

- **tcpdump** is a command-line tool for network traffic analysis.
- It can capture packets from different interfaces (Ethernet, Wi-Fi, etc.) and display detailed information about them.
- The tool operates at the packet level and can display headers of various protocols such as TCP, UDP, ICMP, and more.
- tcpdump can be used to identify vulnerabilities, intercept sensitive data (like unencrypted passwords), analyze network traffic patterns, and perform reconnaissance during penetration testing.

Basic tcpdump Syntax

```
tcpdump [options] [expression]
```

- **[options]**: Defines the behavior of tcpdump (e.g., verbosity, output format).
- **[expression]**: Specifies the criteria to capture specific packets (e.g., filtering traffic for a specific IP address or port).

Key tcpdump Options

1. **-i** : Select the network interface to capture from.
   - Example: `tcpdump -i eth0`
2. **-n**: Prevents DNS resolution (shows IPs instead of hostnames).
   - Example: `tcpdump -n`
3. **-v, -vv, -vvv**: Increase verbosity for packet details.
   - Example: `tcpdump -vvv`
4. **-c** : Capture a specific number of packets.
   - Example: `tcpdump -c 10`

5. **-w** : Write the captured packets to a file (in pcap format) for later analysis.
   - Example: `tcpdump -w capture.pcap`
6. **-r** : Read packets from a previously saved pcap file.
   - Example: `tcpdump -r capture.pcap`
7. **-X**: Display packet contents in both hex and ASCII format.
   - Example: `tcpdump -X`
8. **-s** : Set the snap length (max packet size) to capture. Default is 262144 bytes.
   - Example: `tcpdump -s 0` (captures the full packet).
9. **-tttt**: Display timestamps in a readable format.
   - Example: `tcpdump -tttt`

Packet Capture Filters

Filters help to capture specific packets that match your needs, such as by IP address, port number, or protocol.

Common Filters:

- **IP Address Filtering**: Captures packets from/to a specific IP address.
- `tcpdump host <ip-address>`

  Example: `tcpdump host 192.168.1.1`

- **Port Filtering**: Captures packets on a specific port.
- `tcpdump port <port-number>`

  Example: `tcpdump port 80` (captures HTTP traffic)

- **Protocol Filtering**: Captures traffic for specific protocols.
- `tcpdump proto <protocol>`

  Example: `tcpdump proto icmp` (captures ICMP packets)

- **Traffic Between Two Hosts**: Captures traffic between two hosts.
- `tcpdump src <ip1> and dst <ip2>`

  Example: `tcpdump src 192.168.1.1 and dst 192.168.1.2`

Advanced Filters:

- **Logical Operators**:
  - **and**: Logical AND to combine filters.
  - `tcpdump src 192.168.1.1 and dst 192.168.1.2`
  - **or**: Logical OR for combining multiple conditions.
  - `tcpdump port 80 or port 443`
  - **not**: Logical NOT to exclude packets.

   - `tcpdump not port 22`

Example Scenarios for Filters:

- **Capture all HTTP traffic (port 80)**:
- `tcpdump port 80`
- **Capture traffic from a specific host**:
- `tcpdump host 192.168.0.5`
- **Capture traffic to and from a specific network**:
- `tcpdump net 192.168.0.0/24`
- **Capture only TCP traffic**:
- `tcpdump tcp`

Analyzing Traffic for Penetration Testing

Step 1: Reconnaissance

- **Network Discovery**: Use tcpdump to discover devices, services, and ports on the network.
  - Example: Capture ARP requests to identify live hosts.
  - `tcpdump arp`
- **Identifying Open Ports and Services**: By analyzing TCP/IP handshakes (SYN packets), you can determine open ports.
  - Example: Look for SYN packets to discover open ports.
  - `tcpdump 'tcp[tcpflags] & tcp-syn != 0'`

Step 2: Intercepting Traffic

- **Capture unencrypted traffic**: Identify sensitive data in plaintext (such as HTTP passwords, login information).
  - Example: Look for HTTP traffic with usernames and passwords.
  - `tcpdump tcp port 80 -A`
- **SSL/TLS Traffic**: Capture SSL handshake to analyze encrypted traffic (you cannot decrypt it without the private key, but it is useful for identifying potential targets for SSL stripping or man-in-the-middle attacks).

Step 3: Detecting Vulnerabilities

- **Session Hijacking**: Capture and analyze packets from active sessions to look for opportunities to hijack them.
  - Example: Identify unprotected HTTP cookies in the headers.
  - `tcpdump -A 'tcp port 80' | grep "Set-Cookie"`
- **Scanning for Unusual Traffic**: Analyzing traffic patterns for unusual requests (e.g., port scanning, DDoS attack).
  - Example: Detect repeated SYN packets from a single IP address.
  - `tcpdump 'tcp[tcpflags] & tcp-syn != 0' -c 100`

Step 4: Post-Exploitation

- **Tracking Data Exfiltration**: Use tcpdump to monitor suspicious outbound traffic, such as large amounts of data being sent to external addresses.
  - Example: Capture large outbound packets.
  - `tcpdump -i eth0 'tcp[13] & 0x08 != 0' and len > 1000`

Step 5: Analyzing Malicious Traffic

- **Sniffing for Malware or C&C Servers**: Identify communication between compromised machines and command-and-control (C&C) servers.
  - Example: Filter traffic to known C&C servers.
  - `tcpdump host <C&C-IP>`

Saving and Analyzing Captured Packets

- **Saving Traffic to a File**:
- `tcpdump -w capture.pcap`
- **Reading from a File**:
- `tcpdump -r capture.pcap`
- **Analyzing with Wireshark**: After capturing packets with tcpdump, open them with Wireshark for a more detailed graphical interface.

Security Considerations

- **tcpdump** should be used with caution since it can capture sensitive data, including passwords, cookies, and other private information.
- It's essential to follow ethical guidelines and obtain appropriate permissions when conducting penetration testing and using tcpdump.
- Capturing traffic in a shared or public network could be illegal without consent, and using tcpdump to intercept data without permission is a breach of privacy laws in many jurisdictions.

Conclusion

- **tcpdump** is an indispensable tool in penetration testing for network reconnaissance, vulnerability detection, and monitoring network traffic.
- Mastering **tcpdump** can help penetration testers identify hidden vulnerabilities, unencrypted sensitive data, and unauthorized communication patterns.
- While it is an effective tool for packet capture and analysis, its ethical and legal use is paramount in penetration testing exercises.

Wireshark Traffic Filters - A Detailed Guide

Introduction to Wireshark

Wireshark is one of the most popular and powerful graphical network protocol analyzers. It captures packets from a network interface and allows for detailed inspection of the traffic. Wireshark can help analyze the behavior of protocols, detect network issues, and discover security vulnerabilities.

One of the key features that makes Wireshark so powerful is its **filtering capabilities**. Filters allow you to focus on specific traffic patterns, protocols, or IP addresses, making the analysis process much more manageable.

Wireshark Filters Overview

There are two types of filters in Wireshark:

1. **Capture Filters**: Used to filter the packets during the capture process, meaning only packets that match the filter will be recorded.
2. **Display Filters**: Used to filter and view packets after they have been captured, allowing you to focus on the data that interests you.

1. Capture Filters

Capture filters are used when you first start capturing traffic. Only the packets that meet the criteria of the capture filter are saved.

Basic Capture Filters Syntax

Capture filters use **Berkeley Packet Filter (BPF)** syntax. Some of the common examples are:

- **IP Address Filter**: Capture traffic to/from a specific IP address.
- `host <ip-address>`

  Example: `host 192.168.1.1`

- **Network Filter**: Capture traffic to/from a specific network.
- `net <network>`

  Example: `net 192.168.1.0/24`

- **Port Filter**: Capture traffic on a specific port.

- `port <port-number>`

  Example: `port 80` (Capture HTTP traffic)

- **Protocol Filter**: Capture specific protocols, like TCP, UDP, or ICMP.
- `proto <protocol>`

  Example: `proto tcp` or `proto icmp`

- **Logical Operators**: Combine multiple filters with **and**, **or**, and **not**.
  - **AND**: Capture traffic from a specific IP **and** on a certain port.
  - `host 192.168.1.1 and port 80`
  - **OR**: Capture traffic on **either** port 80 or port 443.
  - `port 80 or port 443`
  - **NOT**: Capture traffic **not** on port 22 (SSH).
  - `not port 22`

Examples of Capture Filters:

- **Capture HTTP traffic only**:
- `port 80`
- **Capture TCP traffic to/from a specific host**:
- `host 192.168.1.10`
- **Capture traffic from a specific network**:
- `net 10.0.0.0/8`

Note: Capture filters are applied when starting the capture session, and they are limited to what the tool can capture. You cannot modify the filter once the capture is started.

---

2. Display Filters

Display filters allow you to filter and refine the data after it's been captured. They are more powerful than capture filters because they let you apply complex conditions on various protocol layers (e.g., IP layer, TCP layer, etc.).

Basic Display Filters Syntax

Display filters use Wireshark's custom syntax. They allow you to filter by almost every field and protocol in the capture.

- **Filter by IP Address**: Display packets from a specific source or destination IP.
- `ip.addr == <ip-address>`

  Example: `ip.addr == 192.168.1.1`

- **Filter by Protocol**: Display packets of a specific protocol.
- `<protocol>`

  Example: `http` or `tcp`

- **Filter by Port**: Display packets from a specific source or destination port.
- `tcp.port == <port-number>`

  Example: `tcp.port == 80`

- **Filter by Specific Layer**: You can filter by any layer in the protocol stack (e.g., Ethernet, IP, TCP).
  - IP address filter (source IP):
  - `ip.src == 192.168.1.1`
  - IP address filter (destination IP):
  - `ip.dst == 192.168.1.1`

Logical Operators:

- **AND**: Combine filters using `&&` or `and`.
- `ip.addr == 192.168.1.1 && tcp.port == 80`
- **OR**: Use `||` or `or` for combining multiple conditions.
- `ip.addr == 192.168.1.1 || ip.addr == 192.168.1.2`
- **NOT**: Use `!` or `not` to exclude certain packets.
- `! tcp.port == 22`

Common Display Filters

- **Filter by Protocol**:
  - Show only HTTP traffic:
  - `http`
  - Show only TCP traffic:
  - `tcp`
  - Show only ICMP (ping) traffic:
  - `icmp`
- **Filter by IP Address**:
  - Show packets from a specific IP:
  - `ip.src == 192.168.1.1`
  - Show packets to a specific IP:
  - `ip.dst == 192.168.1.1`
- **Filter by Port**:
  - Show HTTP traffic (port 80):
  - `tcp.port == 80`
  - Show HTTPS traffic (port 443):
  - `tcp.port == 443`
  - Show traffic on any port greater than 1024:
  - `tcp.port > 1024`
- **Filter by TCP Flags**:
  - Show SYN packets (used for TCP handshakes):

    - `tcp.flags.syn == 1 and tcp.flags.ack == 0`
    - Show FIN packets (indicating session termination):
    - `tcp.flags.fin == 1`
- **Combine Multiple Filters**:
    - Show HTTP traffic from a specific host:
    - `http && ip.src == 192.168.1.1`
    - Show packets that are either HTTP or DNS:
    - `http or dns`

Useful Display Filter Examples for Penetration Testing

- **Capture HTTP Request and Responses**:
- `http.request or http.response`
- **Filter for Specific HTTP Methods**:
    - Show only HTTP GET requests:
    - `http.request.method == "GET"`
- **Capture Specific DNS Queries**:
- `dns.qry.name == "example.com"`
- **Identify Login Attempts** (HTTP POST with credentials):
- `http.request.method == "POST" and tcp.port == 80`
- **Show TCP Handshakes** (SYN and ACK flags):
- `tcp.flags.syn == 1 and tcp.flags.ack == 0`
- **Show TCP Retransmissions** (useful for detecting network issues):
- `tcp.analysis.retransmission`

Tips for Using Filters Effectively

- **Use the Filter Bar**: The filter bar at the top of the Wireshark interface lets you quickly apply display filters. As you type, Wireshark will help auto-complete valid filter expressions.
- **Apply Multiple Filters**: Don’t hesitate to combine multiple filters to narrow down your analysis. You can even use filters for different protocols at the same time.
- **Filter by Packet Length**: You can filter packets based on their size, which is useful when you’re looking for anomalous or malicious traffic.
- `frame.len > 1500`
- **Use the "Follow TCP Stream" Option**: This feature allows you to view the entire conversation between two endpoints. Right-click on a packet, select "Follow" → "TCP Stream."
- **Save and Export Filters**: You can save custom filters for future use. To create a filter in Wireshark, click "Analyze" → "Display Filters" → "Save."

Conclusion

Wireshark’s filtering capabilities are indispensable for detailed network analysis. Understanding both capture and display filters will help you focus on the traffic that matters most during network diagnostics, penetration testing, and security audits. Mastering filters can significantly streamline your process, allowing for efficient and effective packet analysis.

This guide provides a comprehensive overview of Wireshark's filtering features, which will help streamline your network analysis and penetration testing efforts.