SUBMISSION BY – ANKIT SINGH

2201AI47

Tracert Utility Analysis & Advanced Scapy Tracert Utility

Objective:

In this assignment, you will analyze the tracert utility, a fundamental network diagnostic tool. You will explore its functionality, usage, and output, and demonstrate your understanding through a series of tasks.

Tasks:

1. Tracert Basics

- Explain the purpose of the tracert utility and its basic syntax.

The tracert (short for "trace route") utility is a network diagnostic tool used to track the path that data packets take from your computer to a destination host, such as a website or another computer on a network. It is commonly used to identify where a connection may be failing or slowing down.

Syntax : tracert [options] target

- Provide examples of how to use tracert to trace the route to a website and a local host.

```
C:\Windows\System32>tracert google.com

Tracing route to google.com [142.250.194.142]
over a maximum of 30 hops:

  1     7 ms    23 ms    11 ms  10.15.6.1
  2     8 ms   367 ms     7 ms  172.29.1.17
  3    31 ms    14 ms    14 ms  172.16.0.22
  4     *       11 ms    12 ms  14.139.194.1
  5    15 ms     9 ms     6 ms  10.118.248.49
  6     *      249 ms    70 ms  172.31.251.85
  7     *        *       18 ms  172.31.251.84
  8   294 ms    18 ms     *     136.232.74.101
  9     *        *        *     Request timed out.
 10    34 ms     *       29 ms  10.119.234.162
 11     *        *        *     Request timed out.
 12    60 ms    65 ms    79 ms  142.251.226.85
 13    80 ms   231 ms   234 ms  142.251.52.203
 14    74 ms   368 ms    66 ms  del12s05-in-f14.1e100.net [142.250.194.142]

Trace complete.

C:\Windows\System32>
```

2. Tracert Output Analysis

- Run the command tracert (link unavailable) and capture the output.

```
C:\Windows\System32>tracert dfdsf.cc
Unable to resolve target system name dfdsf.cc.

C:\Windows\System32>
```

- Repeat the process for a local host (e.g., tracert 127.0.0.1).

```
PS C:\Users\Ankit Singh\Desktop>  tracert 127.0.0.1

Tracing route to kubernetes.docker.internal [127.0.0.1]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  kubernetes.docker.internal [127.0.0.1]

Trace complete.
PS C:\Users\Ankit Singh\Desktop>
```

3. Tracert Options

- Research and explain the following tracert options:

--d (do not resolve hostnames)

The -d option prevents tracert (Windows) or traceroute (Linux) from resolving IP addresses to hostnames. This can speed up the command because it skips the DNS lookup process for each hop.

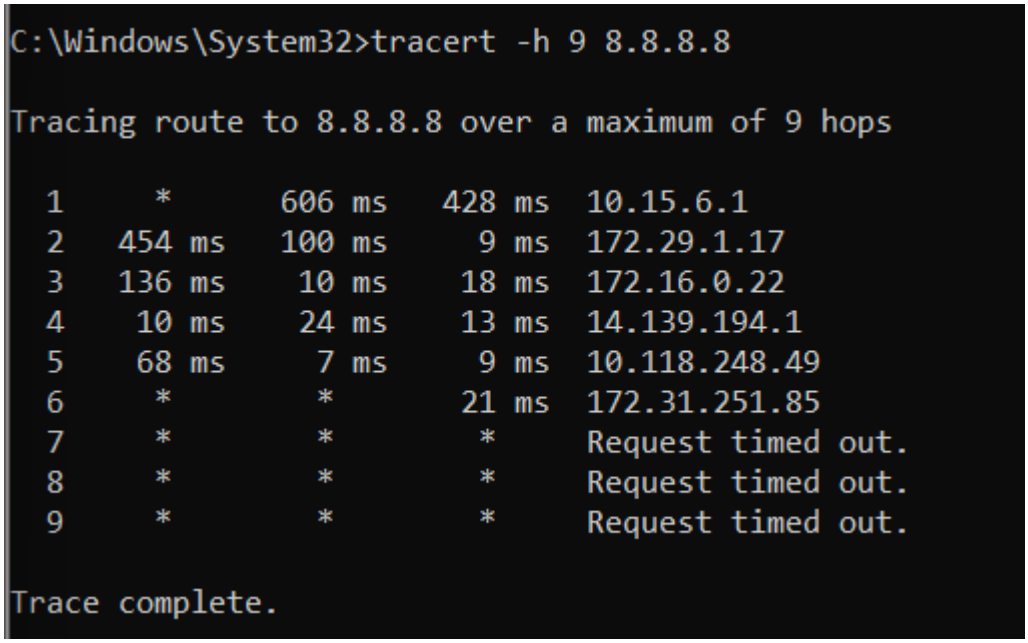
- **Windows:** tracert -d [target]
- **Linux/macOS:** traceroute -n [target]

```
C:\Windows\System32>tracert -d google.com

Tracing route to google.com [142.250.192.206]
over a maximum of 30 hops:

  1     8 ms    21 ms    19 ms  10.15.6.1
  2   310 ms     7 ms     9 ms  172.29.1.17
  3     9 ms     7 ms     6 ms  172.16.0.22
  4     7 ms     8 ms     7 ms  122.252.251.241
  5     9 ms    12 ms    17 ms  122.252.251.197
  6    21 ms    17 ms     *     172.31.251.85
  7     *       17 ms     *     172.31.251.84
  8   132 ms    21 ms     *     136.232.74.101
  9     *        *        *     Request timed out.
 10    30 ms     *        *     10.119.234.162
 11   308 ms    55 ms    79 ms  72.14.194.160
 12    67 ms    48 ms    52 ms  192.178.80.159
 13    60 ms    59 ms    58 ms  142.250.236.55
 14    69 ms   422 ms    62 ms  142.250.192.206

Trace complete.

C:\Windows\System32>
```

--h (maximum number of hops)

The -h option specifies the maximum number of hops the trace will use before stopping. This limits the number of routers (hops) that tracert or traceroute will go through to reach the destination.

- **Windows:** tracert -h [max_hops] [target]
- **Linux/macOS:** traceroute -m [max_hops] [target]



Explanation:

- This command will stop tracing after 10 hops, regardless of whether the destination is reached.

--w (timeout in milliseconds)

The -w option sets the timeout period in milliseconds to wait for each reply from a hop. If a hop doesn't respond within the specified time, it moves on to the next one.

- **Windows:** tracert -w [timeout] [target]
- **Linux/macOS:** traceroute -w [timeout] [target]

```
C:\Windows\System32>tracert -w 3000 8.8.8.8

Tracing route to dns.google [8.8.8.8]
over a maximum of 30 hops:

  1    15 ms    23 ms    24 ms  10.15.6.1
  2    57 ms    40 ms     9 ms  172.29.1.17
  3     8 ms    14 ms     7 ms  172.16.0.22
  4    15 ms     8 ms    11 ms  14.139.194.1
  5    24 ms    22 ms    20 ms  ws197-251-252-122.rcil.gov.in [122.252.251.197]
  6   386 ms    12 ms    15 ms  172.31.251.85
  7   176 ms     *        *     172.31.251.84
  8     *        *        *     Request timed out.
  9     *        *        *     Request timed out.
 10     *       35 ms     *     10.119.234.162
 11     *      199 ms    57 ms  72.14.194.160
 12     *        *        *     Request timed out.
 13   182 ms    51 ms    59 ms  142.251.52.221
 14   135 ms    40 ms    47 ms  172.31.222.243
 15    69 ms    40 ms    92 ms  dns.google [8.8.8.8]

Trace complete.
```

Explanation:

- This command will wait up to 5000 milliseconds (5 seconds) for each hop's response. If a response is not received within that time, the hop will be marked with a *.

- Provide examples of how to use each option.

Example:

- **Windows:** tracert -h 10 google.com
- **Linux/macOS:** traceroute -m 10 google.com

Example:

- **Windows:** tracert -w 5000 google.com
- **Linux/macOS:** traceroute -w 5 google.com (Note: In Linux, the timeout is often in seconds, not milliseconds.)

Example:

- **Windows:** tracert -d google.com
- **Linux/macOS:** traceroute -n google.com

4. Troubleshooting with Tracert

- Describe a scenario where tracert would be used for network troubleshooting (e.g., connectivity issues, slow network speeds).

Scenario: Diagnosing Slow Access to a Website

Notice that accessing a specific website, client-portal.com, is slow, while other sites load quickly. To troubleshoot, you use the tracert (Windows) or traceroute (Linux/macOS) command.

Steps:

1. **Run the Command:**
    - **Windows:** tracert client-portal.com
    - **Linux/macOS:** traceroute client-portal.com
2. **Analyze the Results:**
    - **Local Network Issue:** If delays or timeouts occur in the first few hops, the problem might be with your local network (e.g., router issues).
    - **ISP Issue:** If the issue appears mid-route, it could be with your ISP.
    - **Remote Server Issue:** If the delays occur near the last hops, the problem is likely with the website's server or network.

- Explain how to use tracert to diagnose the issue, including which options to use and why.

To diagnose network issues using tracert (Windows) or traceroute (Linux/macOS), you can use specific options to better understand the path and identify where problems may occur. Here's how to use tracert to diagnose issues:

1. Basic Command Usage

- **Windows:** tracert [target]
- **Linux/macOS:** traceroute [target]

2. Key Options and Their Uses

1. **-d (Do Not Resolve Hostnames)**
    - **Purpose:** Speeds up the command by skipping DNS lookups for hostnames.
    - **Usage:**
        - **Windows:** tracert -d [target]
        - **Linux/macOS:** traceroute -n [target]
    - **Why:** It reduces delay caused by resolving IP addresses to domain names, providing a faster view of the raw route.
2. **-h (Maximum Number of Hops)**
    - **Purpose:** Limits the number of hops tracert will trace.

- **Usage:**
  - **Windows:** tracert -h [max_hops] [target]
  - **Linux/macOS:** traceroute -m [max_hops] [target]
- **Why:** Useful to restrict the trace to a certain number of hops, especially if you suspect the issue might be further down the path.

3. **-w (Timeout in Milliseconds)**
   - **Purpose:** Sets the maximum time to wait for a response from each hop.
   - **Usage:**
     - **Windows:** tracert -w [timeout] [target]
     - **Linux/macOS:** traceroute -w [timeout] [target] (Note: Linux usually uses seconds, not milliseconds.)
   - **Why:** Helps to manage delays and see if any hops are not responding within a reasonable time frame.

5. Conclusion

- Summarize your understanding of the tracert utility and its applications.

The tracert utility (or traceroute on Linux/macOS) is a network diagnostic tool used to trace the path that data packets take from your computer to a destination, such as a website or another network device. It helps identify network issues by showing the route and response times of each hop (router) along the path.

Key Points:

- **Purpose:**
  - **Route Mapping:** Shows each router (hop) the packets pass through to reach the destination.
  - **Troubleshooting:** Identifies where delays or failures occur in the network.
  - **Latency Measurement:** Measures the time it takes for packets to travel to each hop.
- **Basic Usage:**
  - **Command:** tracert [target] (Windows) or traceroute [target] (Linux/macOS)
  - **Output:** Lists each hop with IP addresses or hostnames and response times.
- **Common Options:**
  - **-d (Do Not Resolve Hostnames):** Skips DNS lookups to speed up the trace.

- **-h (Maximum Number of Hops):** Limits the number of hops to trace.
- **-w (Timeout in Milliseconds):** Sets the time to wait for each hop's response.

Applications:

- **Connectivity Issues:** Helps identify if there are problems with reaching a website or server.
- **Network Bottlenecks:** Shows where delays or slowdowns occur along the route.
- **ISP and Routing Issues:** Assists in diagnosing problems with your Internet Service Provider or intermediate network routers.

- Discuss any limitations or potential issues with using tracert for network diagnostics.

While tracert (Windows) and traceroute (Linux/macOS) are valuable tools for network diagnostics, they have limitations and potential issues that users should be aware of:

1. Limited Scope:

- **Single Path Analysis:** tracert shows the path packets take at a given moment but does not account for possible changes in routing or network paths over time.
- **Does Not Diagnose Application Issues:** It only traces network routes and cannot diagnose issues related to specific applications or services.

2. Router Configurations:

- **Firewalls and Filters:** Some routers and firewalls are configured to block or limit ICMP packets used by tracert, leading to incomplete traces or timeouts.
- **TTL (Time-to-Live) Exceeded:** Routers might drop or ignore packets with TTL values that exceed their configured limits, leading to missing hops or incomplete traces.

3. Network Complexity:

- **Multiple Paths:** Large or complex networks might use multiple paths for data, and tracert only shows one path at a time, which might not fully represent network behavior.
- **Load Balancing:** Load balancing can alter the path of packets dynamically, causing variations in tracert results over time.

4. Accuracy and Reliability:

- **Inconsistent Results:** Network congestion or temporary issues might cause inconsistent tracert results, making it difficult to pinpoint the exact problem.
- **Delays and Timeouts:** Variability in response times or timeouts can be misleading, as they might reflect transient issues or network load rather than persistent problems.

5. Security Considerations:

- **Exposure of Network Details:** Using tracert can expose information about the network's topology, which might be sensitive or of interest to malicious actors.

6. Local Network vs. External Issues:

- **Local Issues:** If the problem is within the local network (e.g., a malfunctioning router or switch), tracert may not provide clear indications if the issue occurs before reaching external hops.
- **External Issues:** Problems with external networks or ISPs may not be fully diagnosable through tracert alone, especially if the external network is not responding to ICMP requests.

Objective 2 :

Detailed Report: Scapy-based Traceroute Implementation

1. Additional Features Description:

Features Implemented:

- **Multiple Pings Per Hop:** Allows users to specify how many pings should be sent to each hop for better analysis.
- **Delay Between Pings:** Introduces a configurable delay between consecutive pings, providing better pacing and reducing load on the network.
- **Packet Size Customization:** Allows users to define the size of ICMP packets, with adjustments made for the 28-byte ICMP header.
- **Timeout Control:** Lets users specify how long to wait for a ping reply before moving to the next attempt.

2. How to Use the Features:

To trace a route using the newly implemented features:

Example 1:
Trace a route to google.com :

```
Enter the destination (IP or hostname): google.com
Enter the max TTL (default 30): 30
Enter the number of pings per hop (default 3): 3
Enter the packet size in bytes (default 64): 64
Enter the timeout per ping in seconds (default 2): 2
Enter the delay between pings in seconds (default 1): 1
Traceroute to google.com (216.58.196.110), 30 hops max.
```

Example 2:
Trace a route to 8.8.8.8 using all default parameters:

```
Enter the destination (IP or hostname): 8.8.8.8
Enter the max TTL (default 30): 30
Enter the number of pings per hop (default 3): 3
Enter the packet size in bytes (default 64): 64
Enter the timeout per ping in seconds (default 2): 2
Enter the delay between pings in seconds (default 1): 1
Traceroute to 8.8.8.8 (8.8.8.8), 30 hops max.
```

3. Error Handling Explanation:

1. **Invalid Destination IP/Hostname:**
    - Uses Python's socket.gethostbyname() to resolve the hostname. If the destination cannot be resolved, the error is caught and printed.
    - **Example:** If the user inputs invalidhostname, the output would be:

```
Enter the destination (IP or hostname): ferfe
Enter the max TTL (default 30): 30
Enter the number of pings per hop (default 3): 2
Enter the packet size in bytes (default 64): 64
Enter the timeout per ping in seconds (default 2): 2
Enter the delay between pings in seconds (default 1): 1
Invalid destination IP: [Errno 11001] getaddrinfo failed
```

2. **Handling Ping Failures:**
    - If no reply is received within the timeout, a "Request timed out" message is displayed for each failed ping. Packet loss is calculated based on how many pings were successful.

3. **General Exception Handling:**
    - A try-except block captures any unexpected errors during execution and prints an error message.

4. Sample Output:

Sample 1: Tracing route to google.com with custom parameters.

```
Hop 1:
  Ping 1: 10.15.6.1, RTT: 33.91 ms
  Ping 2: 10.15.6.1, RTT: 98.99 ms
  Ping 3: 10.15.6.1, RTT: 65.41 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 66.11 ms

Hop 2:
  Ping 1: 172.29.1.17, RTT: 67.05 ms
  Ping 2: 172.29.1.17, RTT: 86.00 ms
  Ping 3: 172.29.1.17, RTT: 71.42 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 74.82 ms

Hop 3:
  Ping 1: 172.16.0.22, RTT: 336.43 ms
  Ping 2: 172.16.0.22, RTT: 116.99 ms
  Ping 3: 172.16.0.22, RTT: 123.83 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 192.42 ms

Hop 4:
  Ping 1: 14.139.194.1, RTT: 129.72 ms
  Ping 2: 14.139.194.1, RTT: 26.00 ms
  Ping 3: 14.139.194.1, RTT: 71.64 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 75.79 ms

Hop 5:
  Ping 1: 10.118.248.49, RTT: 69.10 ms
  Ping 2: 10.118.248.49, RTT: 605.36 ms
  Ping 3: 10.118.248.49, RTT: 113.26 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 262.57 ms

Hop 6:
  Ping 1: Request timed out
  Ping 2: Request timed out
  Ping 3: Request timed out
  Summary: 0/3 pings received, Packet loss: 100.00%

Hop 7:
  Ping 1: Request timed out
  Ping 2: Request timed out
  Ping 3: Request timed out
  Summary: 0/3 pings received, Packet loss: 100.00%
```

Sample 2: Tracing route to 8.8.8.8 with default parameters.

```
Hop 1:
  Ping 1: 10.15.6.1, RTT: 37.01 ms
  Ping 2: 10.15.6.1, RTT: 46.99 ms
  Ping 3: 10.15.6.1, RTT: 58.15 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 47.38 ms

Hop 2:
  Ping 1: 172.29.1.17, RTT: 70.79 ms
  Ping 2: 172.29.1.17, RTT: 346.73 ms
  Ping 3: 172.29.1.17, RTT: 115.61 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 177.71 ms

Hop 3:
  Ping 1: 172.16.0.22, RTT: 116.21 ms
  Ping 2: 172.16.0.22, RTT: 120.88 ms
  Ping 3: 172.16.0.22, RTT: 17.00 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 84.70 ms

Hop 4:
  Ping 1: 122.252.251.241, RTT: 71.42 ms
  Ping 2: 122.252.251.241, RTT: 76.00 ms
  Ping 3: 122.252.251.241, RTT: 82.18 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 76.54 ms

Hop 5:
  Ping 1: Request timed out
  Ping 2: Request timed out
  Ping 3: 122.252.251.197, RTT: 1464.59 ms
  Summary: 1/3 pings received, Packet loss: 66.67%, Avg RTT: 1464.59 ms

Hop 6:
  Ping 1: 172.31.251.85, RTT: 149.98 ms
  Ping 2: 172.31.251.85, RTT: 91.83 ms
  Ping 3: 172.31.251.85, RTT: 25.97 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 89.26 ms

Hop 7:
  Ping 1: 172.31.251.84, RTT: 94.18 ms
  Ping 2: 172.31.251.84, RTT: 108.93 ms
  Ping 3: 172.31.251.84, RTT: 112.05 ms
  Summary: 3/3 pings received, Packet loss: 0.00%, Avg RTT: 105.06 ms

Hop 8:
  Ping 1: Request timed out
  Ping 2: Request timed out
```