Malware Traffic Analysis using Wireshark (PCAP-Based Investigation)

1. Overview

Performed a network forensics investigation using Wireshark to analyze PCAP files and identify indicators of malicious activity. Applied protocol-, IP-, and flag-based display filters to isolate FTP, HTTP, DNS, and TCP traffic associated with abnormal communication patterns and potential host compromise.

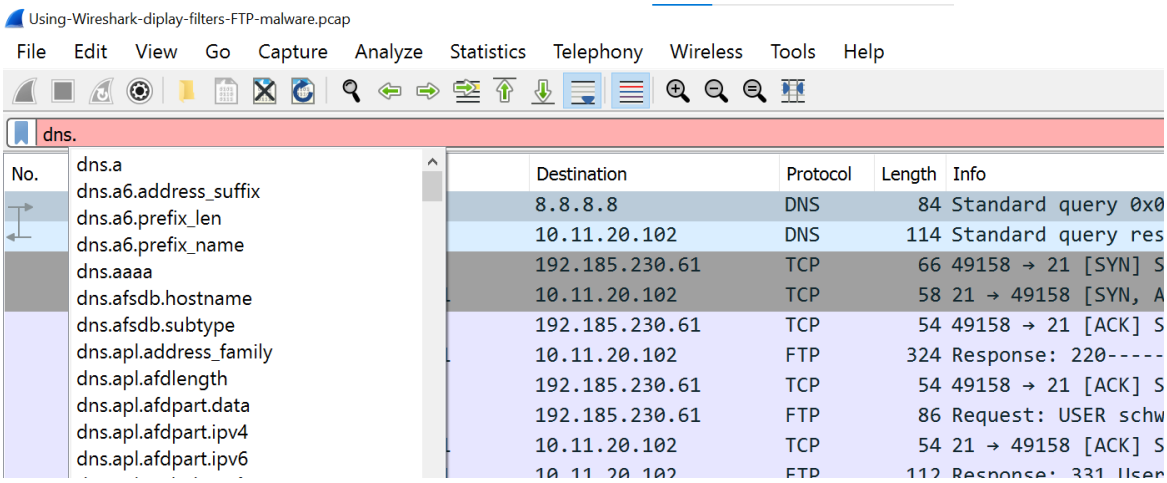
2. Investigation Approach

Wireshark display filters were used to narrow analysis to protocol-specific and host-specific traffic within captured PCAP files. Boolean expressions and protocol fields were applied to efficiently isolate suspicious sessions and analyze communication behavior at the packet level.

3. Filtering Methodology

Two primary filtering approaches were used during analysis:

- **Protocol-based filtering:**
  - Isolated traffic by protocol (DNS, FTP, HTTP, TCP)
  - Leveraged Wireshark autocomplete and protocol-specific fields to refine analysis
- **Address- and field-based filtering:**
  - Applied filters targeting IP addresses, ports, and protocol fields (e.g., ip.addr, tcp.port, udp.port)
  - Used Boolean operators (&&, ||) to correlate traffic across hosts and sessions



Address- and field-based Wireshark display filters (IP address and protocol fields) were used to refine DNS traffic and isolate relevant network activity.

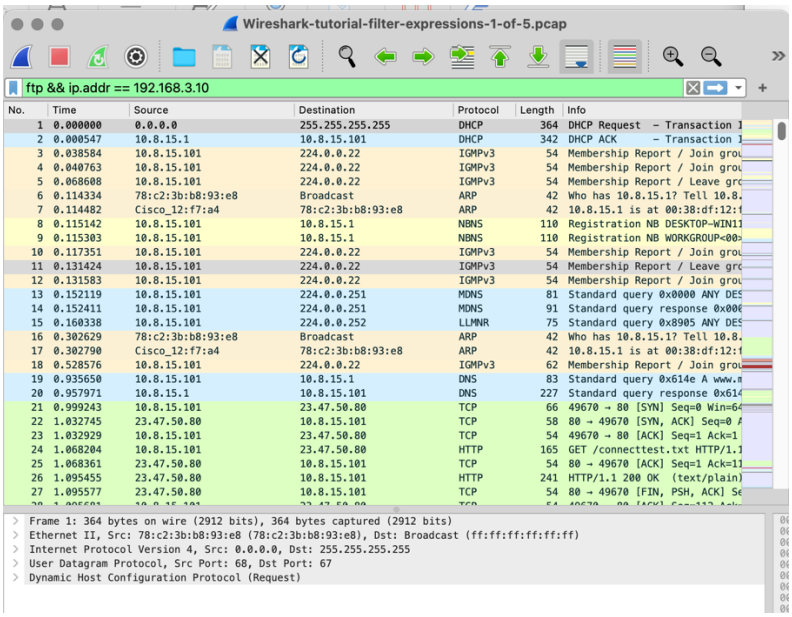
Address- and field-based display filters were used to narrow analysis to specific hosts, ports, and protocols.

4. Web-Based Traffic Analysis

Web-based traffic was analyzed to identify suspicious client behavior and repeated connection attempts indicative of potential compromise. Wireshark display filters were applied to isolate HTTP and related traffic, allowing focused inspection of request patterns and session activity.

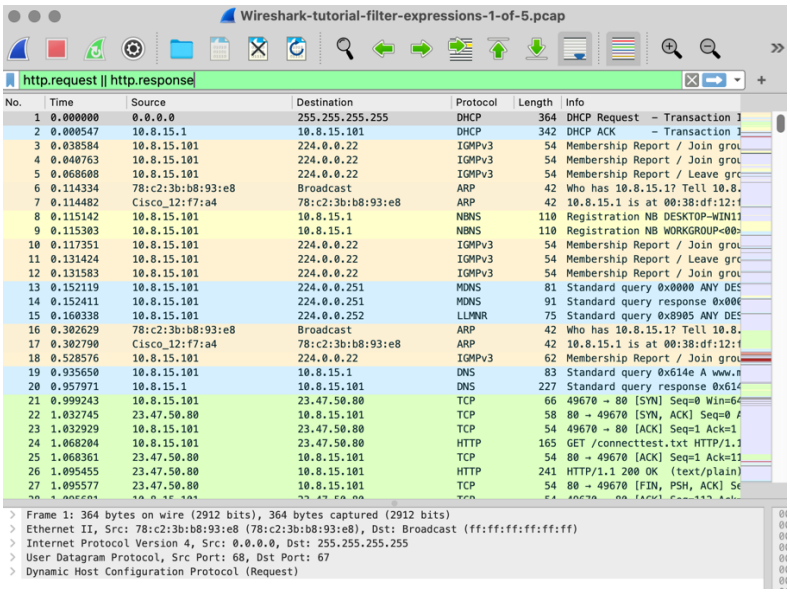
- **Analysis performed:**
    - Applied http.request display filter to isolate outbound HTTP requests initiated by the internal host.
    - Filtered traffic to identify repeated connection attempts and abnormal retransmissions.
    - Observed request patterns consistent with automated or non-user-initiated behavior.

The filtered results revealed active web communication between the internal host and external destinations, supporting further investigation into host behavior and potential compromise.



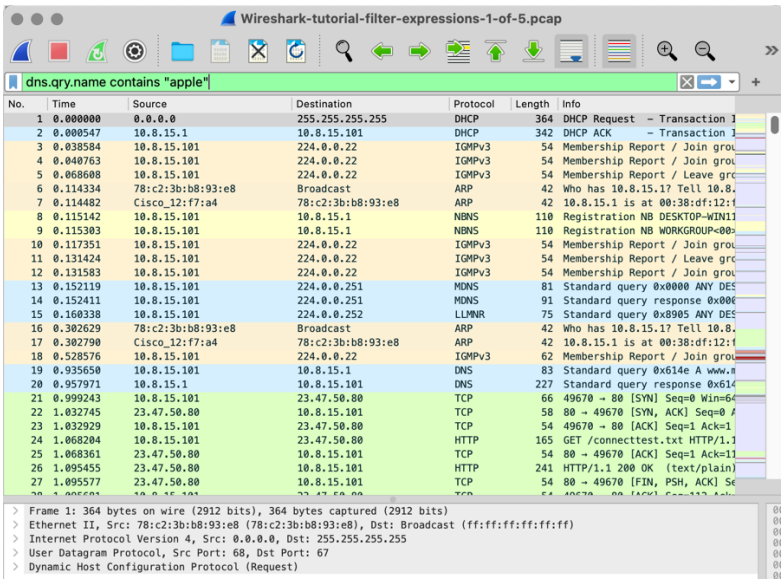
FTP Traffic Analysis

Applied the display filter ftp && ip.addr == 192.168.3.10 to isolate FTP traffic associated with the internal host. The filter successfully returned FTP control and data packets, confirming active FTP communication involving the specified IP address.



HTTP Traffic Analysis

Applied the display filter http.request || http.response to isolate inbound and outbound HTTP communication. The filtered results revealed active HTTP sessions between the internal host and external servers, confirming normal request–response behavior and enabling further session-level inspection.

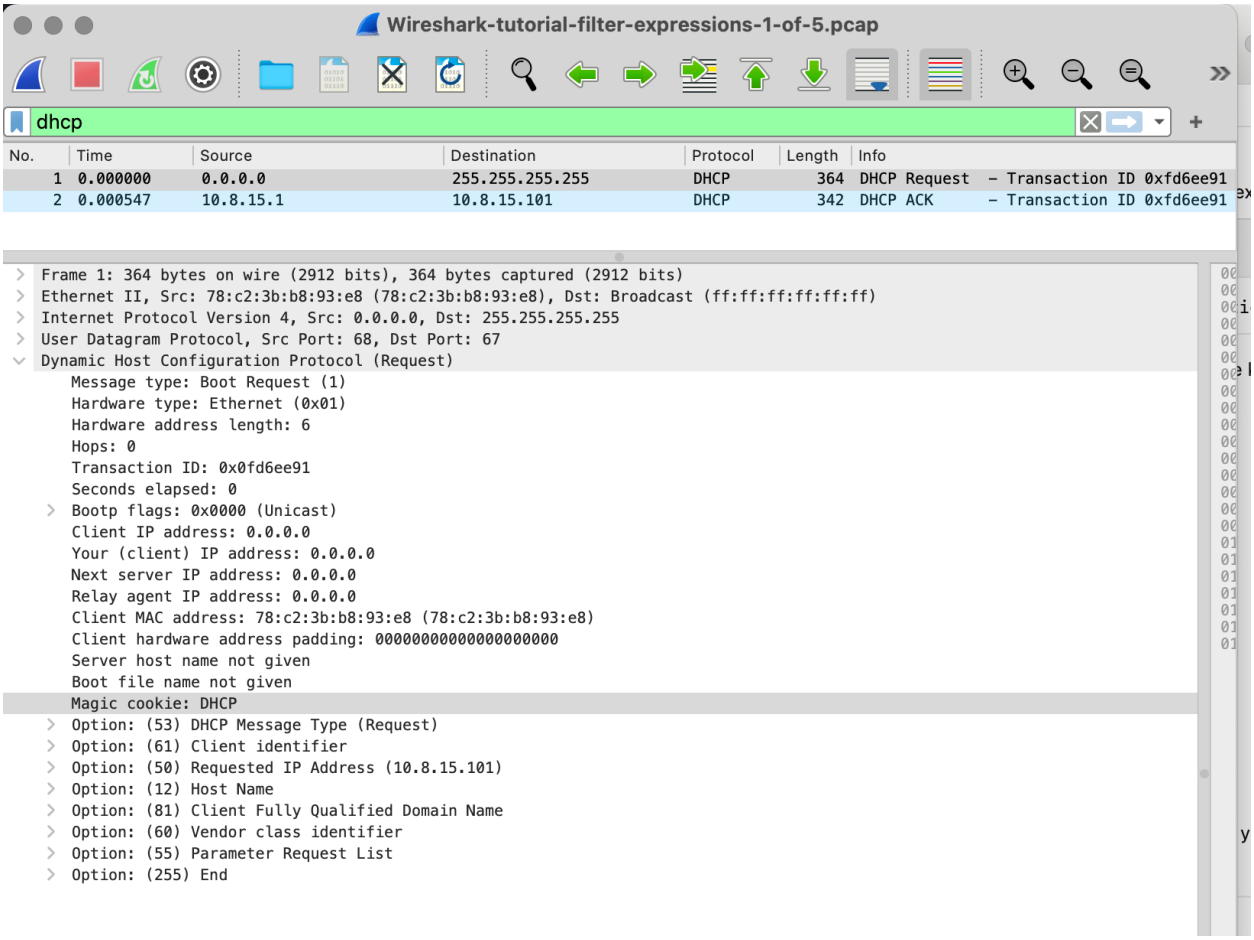


DNS Traffic Analysis

Applied the display filter dns.qry.name contains "apple" to identify DNS queries associated with Apple-related domains. The results confirmed external name resolution activity and helped establish baseline DNS behavior for the host.

5. Identifying Hosts and Users

Step 3: Host name and MAC address

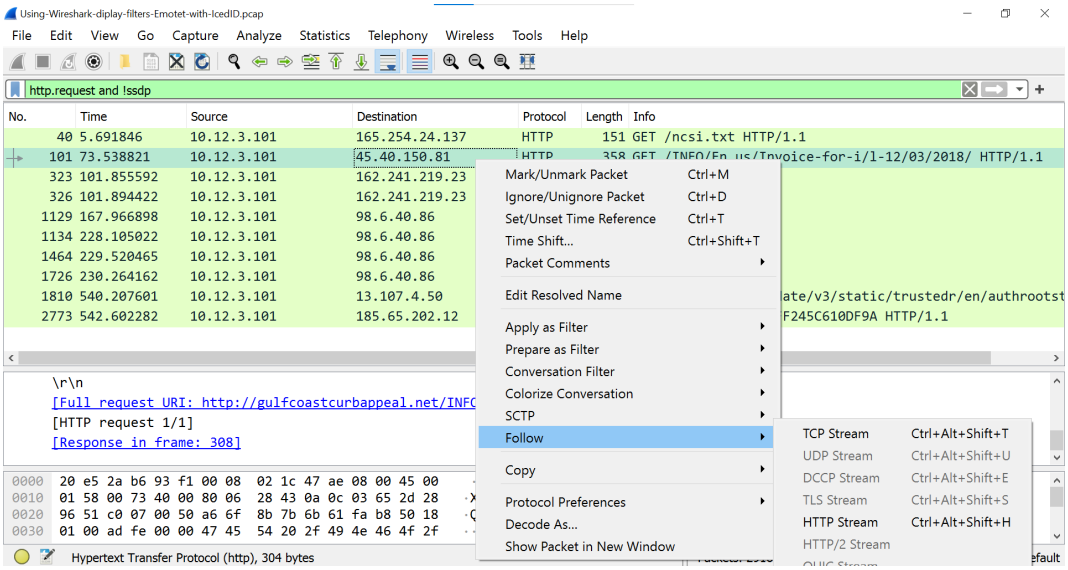


Host Identification via DHCP

DHCP traffic was analyzed to identify the internal host responsible for initiating network activity. Inspection of the DHCP Request packet revealed the client MAC address **78:c2:3b:b8:93:e8**, which was used to associate subsequent traffic with a specific device. No hostname was included in the DHCP options, indicating the host did not advertise a hostname during the request.

6. Operating System

User-agent strings from headers in HTTP traffic can reveal the operating system. If the HTTP traffic is from an Android device, you might also determine the manufacturer and model of the device.



Operating System Identification

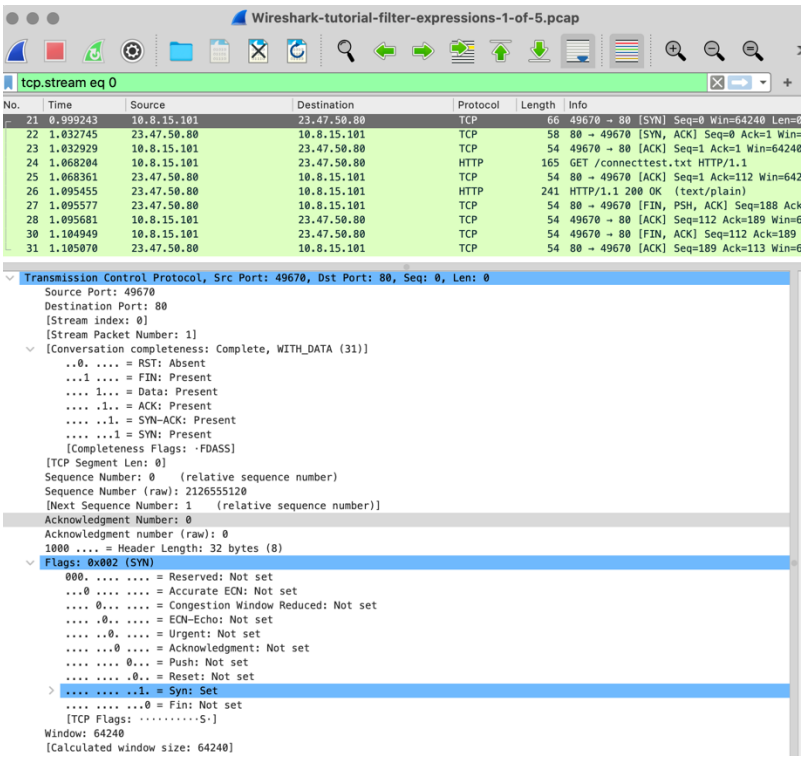
HTTP traffic was analyzed to determine the operating system of the infected host by inspecting the User-Agent header within HTTP requests. The display filter http.request and !(ssdp) was applied to isolate relevant HTTP traffic.

Inspection of the TCP stream associated with destination IP address **23.47.50.80** revealed the following User-Agent string:

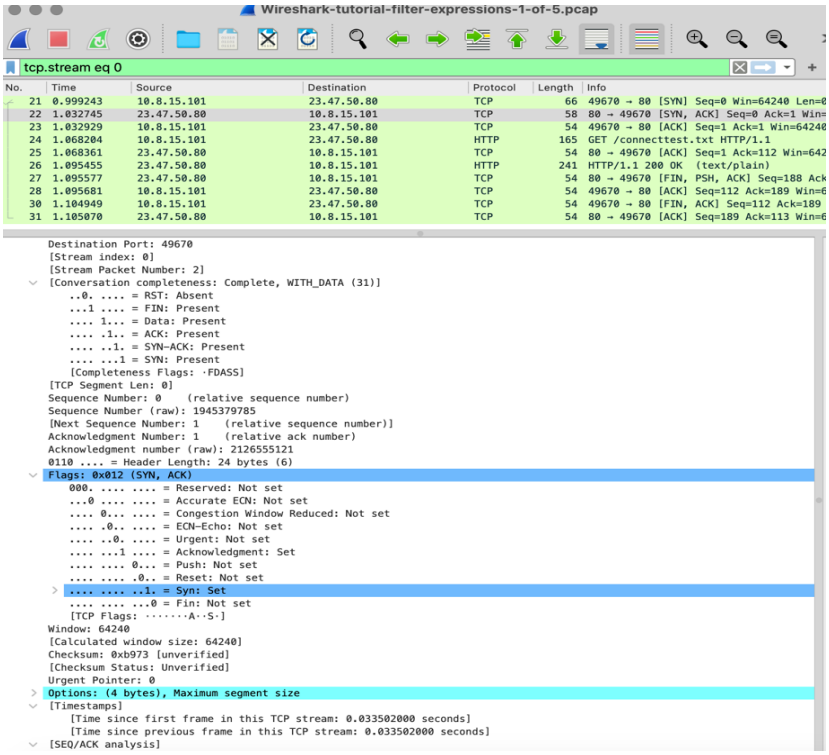
User-Agent: Microsoft NCSI

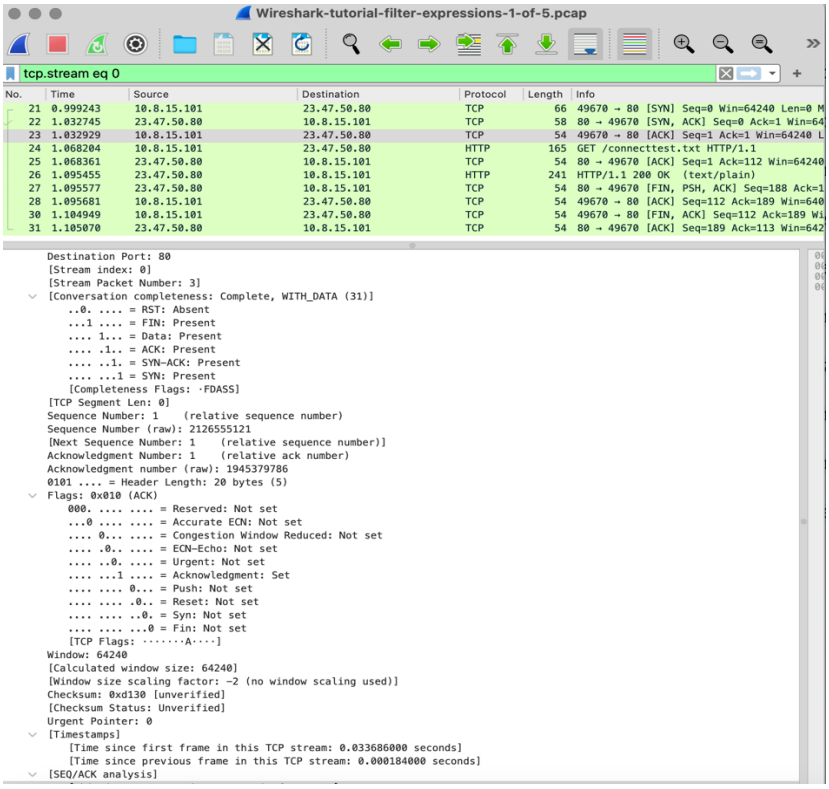
This User-Agent indicates that the host is running a Microsoft Windows operating system, as Microsoft NCSI (Network Connectivity Status Indicator) is used by Windows systems to perform connectivity checks.

Additionally, analysis of the TCP three-way handshake between **10.8.15.101** and **23.47.50.80** confirmed a successful TCP connection establishment, supporting normal HTTP communication behavior between the internal host and the external server.



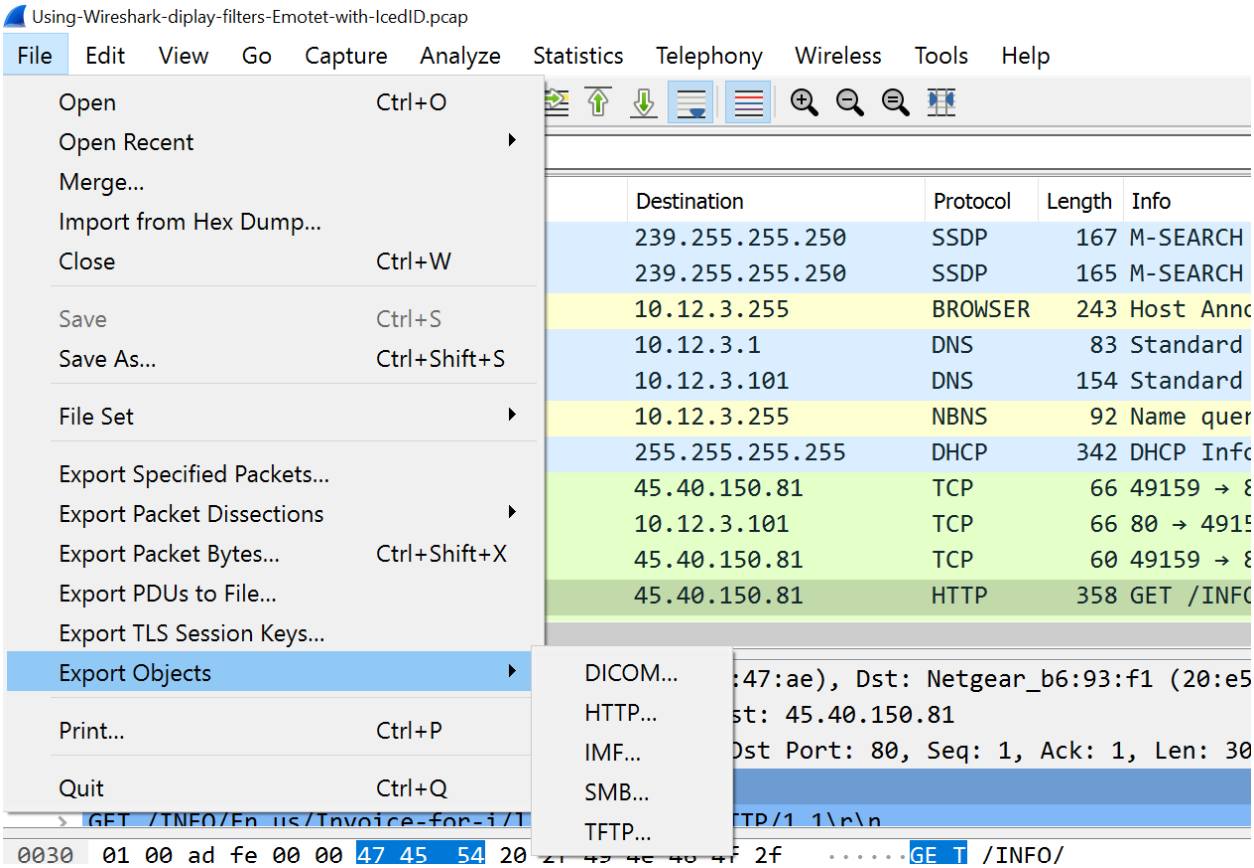
The following screenshots provide supporting packet-level evidence for the operating system identification and TCP connection establishment discussed in Step 4.





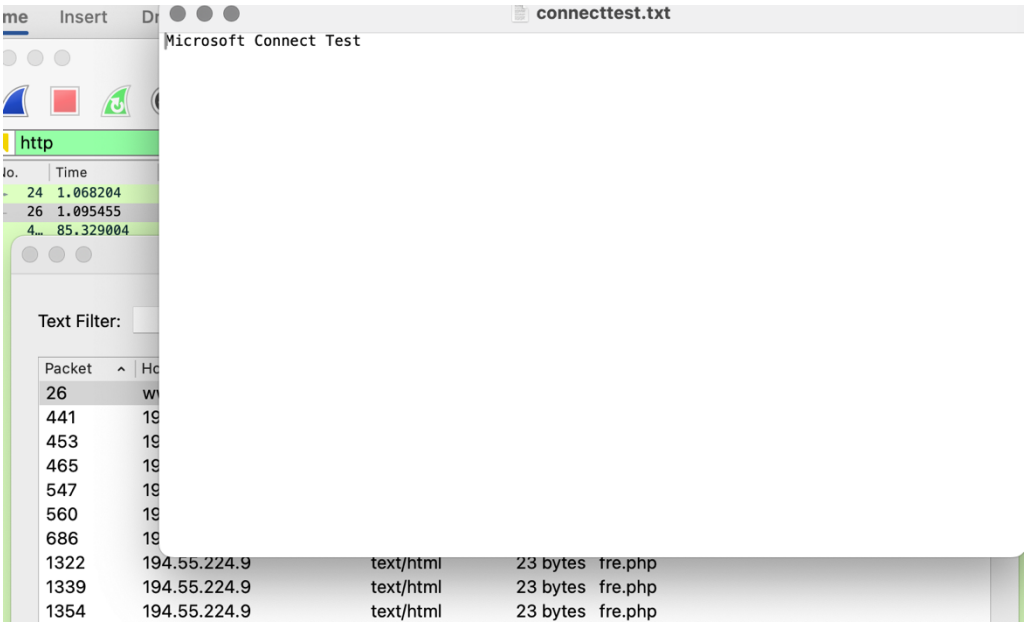
7. **CP Stream and Handshake Evidence:** The screenshots above show the TCP stream and associated flags exchanged between the internal host and the external destination, confirming successful session establishment prior to HTTP data transfer.

8. Export objects from HTTP traffic



9. Extracted File Analysis:

The extracted file, connecttest.txt, contains content associated with Microsoft Network Connectivity Status Indicator (NCSI) checks. This confirms normal Windows connectivity behavior and aligns with the previously identified User-Agent information.



10. Extracted File Preview