TECHNICAL REPORT

Cybersecurity Threat Analysis & Network Packet Investigation

Practical Detection of MITM, DoS, and C2 Attack Patterns Using Wireshark Network Protocol Analyzer

---

Author: Vishal Panda
Institution: VIT Bhopal University
Program: BTech Cybersecurity (Expected 2026)
Email: vishal99.business@gmail.com
GitHub: **github.com/Rolexx90/cybersecurity-threat-analysis**

Project Duration: April - May 2025
Project Context: Security Assessment Internship, House of Couton Pvt. Ltd

---

ABSTRACT

This report documents a comprehensive network security analysis conducted using Wireshark to identify and analyze attack patterns in simulated network traffic. Four distinct attacks were detected and documented: ARP Spoofing (Man-in-the-Middle), SYN Flood (Denial of Service), UDP Flood reconnaissance, and HTTP-based Command & Control traffic. The analysis includes detection methodology, packet-level indicators of compromise, and industry-aligned mitigation strategies.

Keywords: Network Security, Wireshark, Threat Detection, Packet Analysis, MITM Attack, DoS Attack, Command & Control

April - May 2025

Table of Contents

Part A: Research Report – Common Cyber Attacks & Controls

1.1 Man-in-the-Middle (MITM) Attack

An intercepted message between two persons usually involves an intrusion of an attacker when he or she suspectedly or silently relieves the communication between the victims without them being conscious about the interference. The involved attacker impersonates both participants and entraps them into believing that they are truly communicating directly with the partner (Conti et al. 2016).

Some so many techniques that would lead to MITM attacks include ARP spoofing, DNS spoofing, or HTTPS stripping. Alternatively, ARP spoofing facilitates the Address Resolution Protocol table manipulation and reroutes traffic to the attacker's machine, where the eavesdropping or data manipulation occurs.

One of the primary effects of MITM attack is the sensitive information compromise, such as secret logins, financing details, or sensitive corporate communications. Most effective types of attacks occur mostly in unencrypted or public networks.

The interest of attacks known as Man-in-the-Middle has matured beyond the era of ARP spoofing on a LAN. With the enhancement of wireless technologies, attackers are making use of rogue access points or Evil Twin attacks to tempt users into connecting to a malicious hotspot. After being connected to the rogue network, traffic from the user can be intercepted and decrypted either with an SSL strip or with fake certificates, as noted by Gruschka & Iacono (2009).

Emerging Trends:

Use of automated tools such as Ettercap, Bettercap, or WiFi Pineapple has made MITM a streamlined process for attackers.

In corporate environments, email spoofing with MITM is a primary vector for Business Email Compromise (BEC) scams.

Industry Insight: In the 2023 Data Breach Investigations Report of Verizon, it is stated that more than 15% of the breaches have direct applications of interception techniques analogous to MITM especially in the phishing-based campaigns.

Mitigation Controls:

- **Encryption:** Strong end-to-end encryption (e.g., TLS) implementation helps to obviate the possibility of interception and tampering.
- **Authentication Mechanisms:** Mutual authentication and digital certificates should be used.
- **Network Security Tools:** These include IDS and firewalls for monitoring suspicious ARP traffic.
- **VPN Usage:** Encourage employees to make use of VPNs when they are on public networks.

Diagram: MITM via ARP Spoofing *(Insert a simple diagram showing attacker between client and server)*

Real-World Example: A practical case in 2017 affected more users sitting in coffee shops by creating fake Wi-Fi hotspots...Users unknowingly connected, allowing attackers to set up an MITM and steal credentials (Kizza, 2020).

1.2 Denial-of-Service (DoS) Attack

This kind of denial-of-service attack is one that disrupts services or networks by making them so overloaded with traffic or taking in the data that can cause their crashing. This results into blocking the legal public from accessing such resources (Zhou et al., 2018).

Apart from the traditional SYN flooding, modern Denial of Service attacks are mostly characterized by Distributed Denial of Service (DDoS) attacks, which utilize thousands of devices worldwide through coordinated botnets. Such attack events are always available on Attack-as-a-Service platforms, making them even accessible to less-skilled enemies (Rossow, 2014).

Emerging Variants:

- **Application-layer DDoS:** Target APIs or login endpoints.
- **Reflection & Amplification Attacks:** Bounce traffic through open resolvers (e.g., DNS, NTP) to greatly amplify volume.
- **Cloud Mitigation Tools:** Cloud-native services such as AWS Shield or Google Cloud Armor implement scalable mitigation by filtering traffic at the edge before routing it to enterprise systems.

- **Cost Impact report**: DoS attacks have a hence malicious financial impact. Per Kaspersky (2022), DDoS attacks would cost an average over $120,000 for every hour an enterprise remains disabled.

DoS attacks can take various forms, including:

- **Volume-Based Attacks:** UDP flood, ICMP flood.
- **Attacks using protocols:** Taking advantage of weaknesses found in protocol layers. (e.g. SYN floods).
- **Application-Layer Attacks:** Overloading of particular functions. Example is HTTP GET/POST floods.

Mitigation Controls:

- **Rate Limiting:** It restricts number of requests made to critical resources.
- **Traffic Filtering:** Firewalls and routers may filter out the malicious traffic.
- **Redundancy:** Resilience is enhanced by load balancing and failover mechanisms.
- **Use of Anti-DDoS Services:** E.g., Cloud-based mitigation platforms such as Cloudflare or Akamai**.**

Diagram: *(Insert flowchart showing attack traffic overwhelming a server)*

Real-World Example: In 2016, the Mirai botnet was responsible for one of the largest denial-of-service attacks to date, peaking at 1.2 Tbps and affecting such popular websites as Twitter and Netflix (Antonakakis et al., 2017).

1.3 SQL Injection

Basically, SQL Injection (SQLi) refers to a technique of injecting code to take advantage of certain loopholes on the software of an application that takes from the insertion of a malevolent SQL query into the input fields. If the query gets executed, it may divulge, modify, or delete sensitive data in databases (Halfond et al., 2006).

Traditional SQL injection is known for exploiting weak input validation and executing queries like ' OR '1'='1 to access unauthorized data or bypass login forms. SQLi is a commonly heard term used for significant breaches, financial loss, and reputational loss.

Insecure coding skills and ill development lifecycle governance are among the poorer reasons as to why SQLi vulnerabilities are so prevalent. The modern Blind SQL Injection does not show output directly but uses boolean conditions to determine data that are not easily detected.

The SQLi could have been a trigger for several attacks, such as Privilege Escalation or Remote Code Execution, often in conjunction with command injection vulnerabilities.

Security Frameworks:

- SQL Injection regularly finds a place in the OWASP Top 10 as one of the most important web vulnerabilities.

- Practice in Secure SDLC will facilitate code reviews, automated static analysis, and the incorporation of DevSecOps, thus substantially limiting exposure.

Mitigation Controls:

- **Prepared Statements (Parameterized Queries):** prohibit dynamic executing input.
- **Input Validation:** Each and every user input needs to be sanitized.
- Principle of Least Privilege: Restrict on database permissions for the accounts of application.
- **Web Application Firewalls (WAF):** Identify and prevent attempts at SQLi.

Diagram: *(Insert example of SQL injection into login form)*

Real-World Example: The SQLi vulnerability exposed millions of user credentials in the 2012 LinkedIn breach (Barnett, 2013).

1.4 Zero-Day Exploit

When an intruder successfully attacks a software vulnerability that the vendor is not aware of, hence not patched, it is called a zero-day exploit. This window of opportunity is often used by the attackers before a fix is introduced, causing widespread compromise (Bilge & Dumitras, 2012).

It is exactly the very nature of stealth and unpredictability that Zero-Day attacks offer that makes them so dangerous. Threat actors may choose to deliver malware payloads via spear-phishing or malicious links.

In underground markets, zero-day vulnerabilities are prized. Some exploits even fetch millions of dollars, depending on the targeted software (e.g., Windows, iOS). Nation-state actors frequently resort to attacking each other with zero-day attacks in cyber-espionage, particularly targeting their critical infrastructure and government networks.

Lifecycle Stages:

- **Discovery:** Usually happens through threats from actors or security researchers.
- **Exploitation:** Up till being discovered and disclosed.
- **Disclosure:** Mostly via bug bounty programs or by vendors.
- **Patch deployment:** Time lag often leaves systems exposed therefore.

Mitigation Controls:

- **Endpoint Protection:** Use heuristic-based antivirus and EDR type of solutions.
- **Patch Management:** Update systems as frequently as possible to minimize exposure windows.
- **Threat Intelligence:** Put intelligence feeds into use for detection of new threats.
- **Segmentation:** Stop the spread of infection in the broken-down networks.

Diagram: *(Show lifecycle from discovery to patching of zero-day)*

Real-World Example : It was in the year 2010 when this so called Stuxnet came into the limelight. It, having exploited the number of zero-day vulnerabilities in the software application from Siemens, acted as a disruptor to the Iranian nuclear facilities (Langner, 2011).

1.5 DNS Tunneling

In this type of attack, hackers thieve encoded data in DNS queries and responses. The so-called attacks help to get past firewall restrictions and 'illicitly' exfiltrate data. Because DNS is often allowed through firewall traffic, it therefore turns into a stealthy and silent avenue (Sidorov & Sgandurra, 2019). In most cases it facilitates command and control (C2) communications which some forms of malware rely on for accessing long-term entries.

Internally, through DNS tunneling, an attacker might not only exfiltrate data outward but also deploy covert channels within internal networks that would bypass segmentation to initiate C2 operations without detection.

- **Detection Complexity**: Everyday security appliances will often fail to flag DNS tunneling as, in reality, the tunneling traffic conforms to expected DNS formats. Therefore, advanced analytics, utilizing machine learning and entropy-based detection, are increasingly being demanded (Cambiaso et al., 2020).

Example: The APT group known as OilRig extensively used DNS tunneling to ensure persistent access to energy companies in the Middle East, illustrating the usage of such techniques in state-sponsored campaigns.

Evolving Defenses:

- DNS Security Extensions (DNSSEC) provide origin and integrity validation but do not intrinsically prevent tunneling.
- SIEM integration with DNS logs enhances visibility into Security Operations Centers (SOCs).

Mitigation Controls:

- **DNS Traffic Monitoring:** Looking for abnormal query patterns.
- **Security Solutions:** Implement DNS firewall and threat intelligence services.
- **Block External Resolvers:** Prevent unauthorized DNS server usage.
- **Deep Packet Inspection:** Inspect DNS payloads for tunnelling behavior.

Diagram: *(Show DNS query with embedded data payload)*

Real-World Example: In 2018, the "DNSpionage" campaign, which involved the use of DNS tunneling for exfiltration of data, also targeted companies in the Middle East (Cisco Talos, 2018).

Part B: Practical Network Packet Analysis – Wireshark

2.1 Methodology

The packet analysis process was carried out in Wireshark, a network protocol analyzer that can deep-inspect hundreds of protocols. The analysis was done to identify possible patterns of malicious activity, using display filters, packet inspection, and behavior correlation; a .pcap file with simulated attack traffic was analyzed.

Followed Steps:

- **Initial Scanning:** Overall traffic observed, including the number of packets and protocols used.
- **Filter Usage:** Filters, such as arp, tcp.flags.syn==1 && tcp.flags.ack==0, dns, and http.request, were applied to isolate anomalies
- **Behavioral Analysis**: Unusual behaviors of hosts, including repetitious ARP broadcasts, high SYN packet rates, or suspicious DNS queries, were reviewed.
- **Payload Inspection:** Content of packets was inspected for signs of malicious acts or for obfuscated data.
- **Cross-Correlation:** Probed timing and protocol usage to match up host activities and classify them as possible intrusion attempts.

Wireshark's color display helped in visualization, along with its detail pane for each suspicious interaction.

2.2 Attack 1: ARP Spoofing (MITM)

Description:

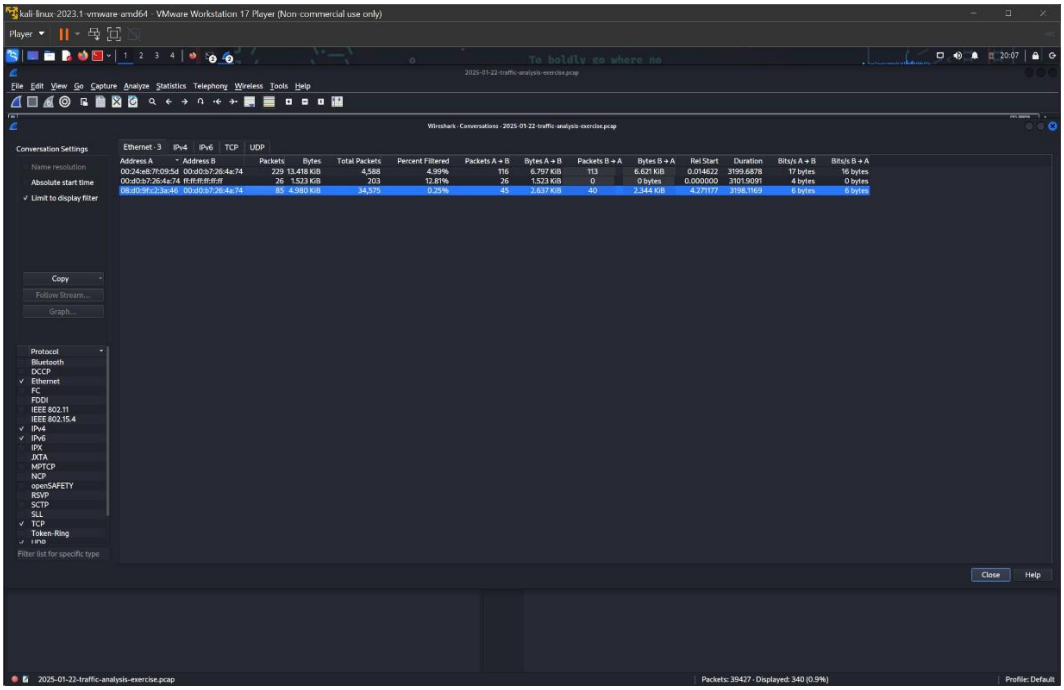
The act of ARP Spoofing involves an individual or hacker sending fake ARP messages in order to link up a MAC address to an IP address of a real host. This is typically done for purposes of intercepting network traffic.

Detection:

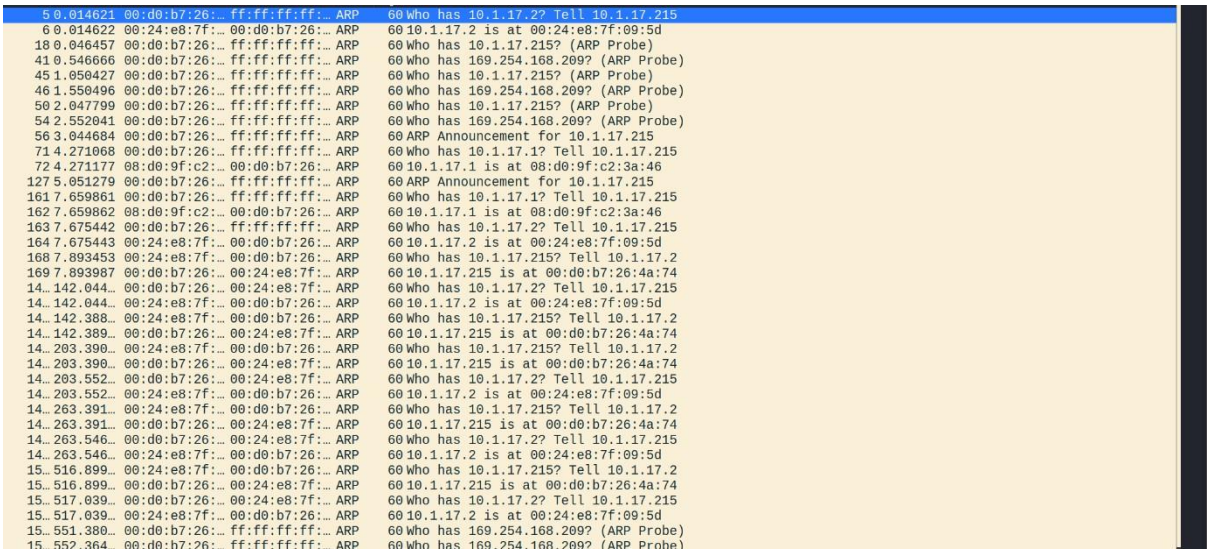
- **Wireshark filter**: arp
- Anomalities observed within the same MAC claim to different IPs through repeated ARP replies.

Packet Details:

- **Opcode:** reply (2)
- Sender MAC: 00:0d:b7:26:4a:74
- Sender IP**:** 10.1.17.215
- Target MAC: 00:24:e8:7f:09:5d
- Target IP**:** 10.1.17.2

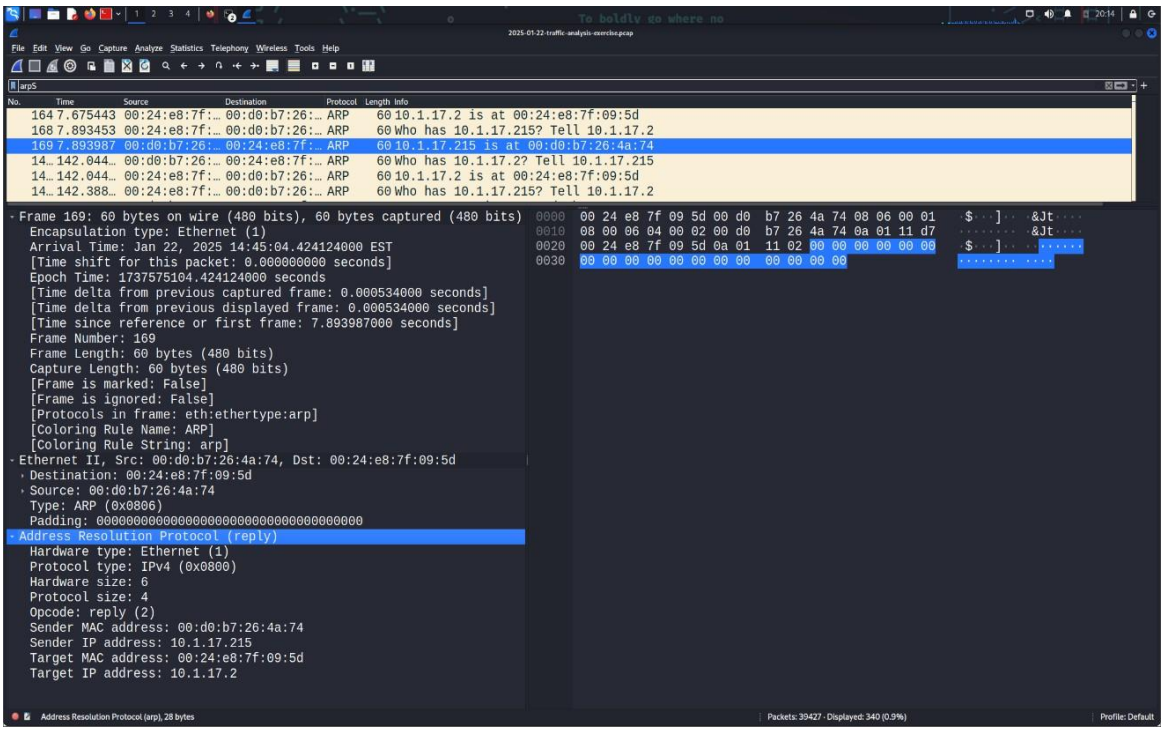


“As can be seen from the Ethernet conversations table, MAC address 00:0d:b7:26:4a:74 is communicating with several peers, including the broadcast address, repeatedly. This is a sign of ARP spoofing activity where the attacker sends spoofed ARP replies to fill up the network with poisoned caches.”



No.	Time	Source	Destination	Protocol	Length	Info
5	0.014621	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 10.1.17.2? Tell 10.1.17.215
6	0.014622	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	10.1.17.2 is at 00:24:e8:7f:09:5d
18	0.046457	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 10.1.17.215? (ARP Probe)
41	0.546666	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 169.254.168.209? (ARP Probe)
45	1.050427	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 10.1.17.215? (ARP Probe)
46	1.550496	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 169.254.168.209? (ARP Probe)
50	2.047799	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 10.1.17.215? (ARP Probe)
54	2.552041	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 169.254.168.209? (ARP Probe)
56	3.044684	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	ARP Announcement for 10.1.17.215
71	4.271068	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 10.1.17.1? Tell 10.1.17.215
72	4.271177	08:d0:9f:c2:…	00:d0:b7:26:…	ARP	60	10.1.17.1 is at 08:d0:9f:c2:3a:46
127	5.051279	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	ARP Announcement for 10.1.17.215
161	7.659861	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 10.1.17.1? Tell 10.1.17.215
162	7.659862	08:d0:9f:c2:…	00:d0:b7:26:…	ARP	60	10.1.17.1 is at 08:d0:9f:c2:3a:46
163	7.675442	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 10.1.17.2? Tell 10.1.17.215
164	7.675443	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	10.1.17.2 is at 00:24:e8:7f:09:5d
168	7.893453	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	Who has 10.1.17.215? Tell 10.1.17.2
169	7.893987	00:d0:b7:26:…	00:24:e8:7f:…	ARP	60	10.1.17.215 is at 00:d0:b7:26:4a:74
14…	142.044…	00:d0:b7:26:…	00:24:e8:7f:…	ARP	60	Who has 10.1.17.2? Tell 10.1.17.215
14…	142.044…	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	10.1.17.2 is at 00:24:e8:7f:09:5d
14…	142.388…	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	Who has 10.1.17.215? Tell 10.1.17.2
14…	142.389…	00:d0:b7:26:…	00:24:e8:7f:…	ARP	60	10.1.17.215 is at 00:d0:b7:26:4a:74
14…	203.390…	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	Who has 10.1.17.215? Tell 10.1.17.2
14…	203.390…	00:d0:b7:26:…	00:24:e8:7f:…	ARP	60	10.1.17.215 is at 00:d0:b7:26:4a:74
14…	203.552…	00:d0:b7:26:…	00:24:e8:7f:…	ARP	60	Who has 10.1.17.2? Tell 10.1.17.215
14…	203.552…	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	10.1.17.2 is at 00:24:e8:7f:09:5d
14…	263.391…	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	Who has 10.1.17.215? Tell 10.1.17.2
14…	263.391…	00:d0:b7:26:…	00:24:e8:7f:…	ARP	60	10.1.17.215 is at 00:d0:b7:26:4a:74
14…	263.546…	00:d0:b7:26:…	00:24:e8:7f:…	ARP	60	Who has 10.1.17.2? Tell 10.1.17.215
14…	263.546…	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	10.1.17.2 is at 00:24:e8:7f:09:5d
15…	516.899…	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	Who has 10.1.17.215? Tell 10.1.17.2
15…	516.899…	00:d0:b7:26:…	00:24:e8:7f:…	ARP	60	10.1.17.215 is at 00:d0:b7:26:4a:74
15…	517.039…	00:d0:b7:26:…	00:24:e8:7f:…	ARP	60	Who has 10.1.17.2? Tell 10.1.17.215
15…	517.039…	00:24:e8:7f:…	00:d0:b7:26:…	ARP	60	10.1.17.2 is at 00:24:e8:7f:09:5d
15…	551.380…	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 169.254.168.209? (ARP Probe)
15…	552.364…	00:d0:b7:26:…	ff:ff:ff:ff:…	ARP	60	Who has 169.254.168.209? (ARP Probe)

“In the intercepted ARP traffic, we noticed repeated announcements and responses mapping multiple IPs (such as 10.1.17.215 and 10.1.17.2) to the same or other MAC addresses (such as 00:0d:b7:26:4a:74). Such conflicting responses, coupled with the abnormal frequency of ARP messages, indicate that an attacker was trying to poison ARP tables of adjacent hosts — a common Man-in-the-Middle (MITM) configuration via ARP Spoofing.”



Observation: ARP table is poisoned with traffic rerouted through the attacker device.

Mitigation:

- Dynamic ARP inspection should be enabled.
- Important systems should be assigned static entries in ARP.

2.3: SYN Flood (DoS Attack)

SYN Flooding is sending a flood of SYN requests to a system until it consumes all of its resources, rendering itself unresponsive.

Detection:

- Wireshark filter: tcp.flags.syn==1 && tcp.flags.ack==0
- Other SYN requests were sent without the presence of ACK.

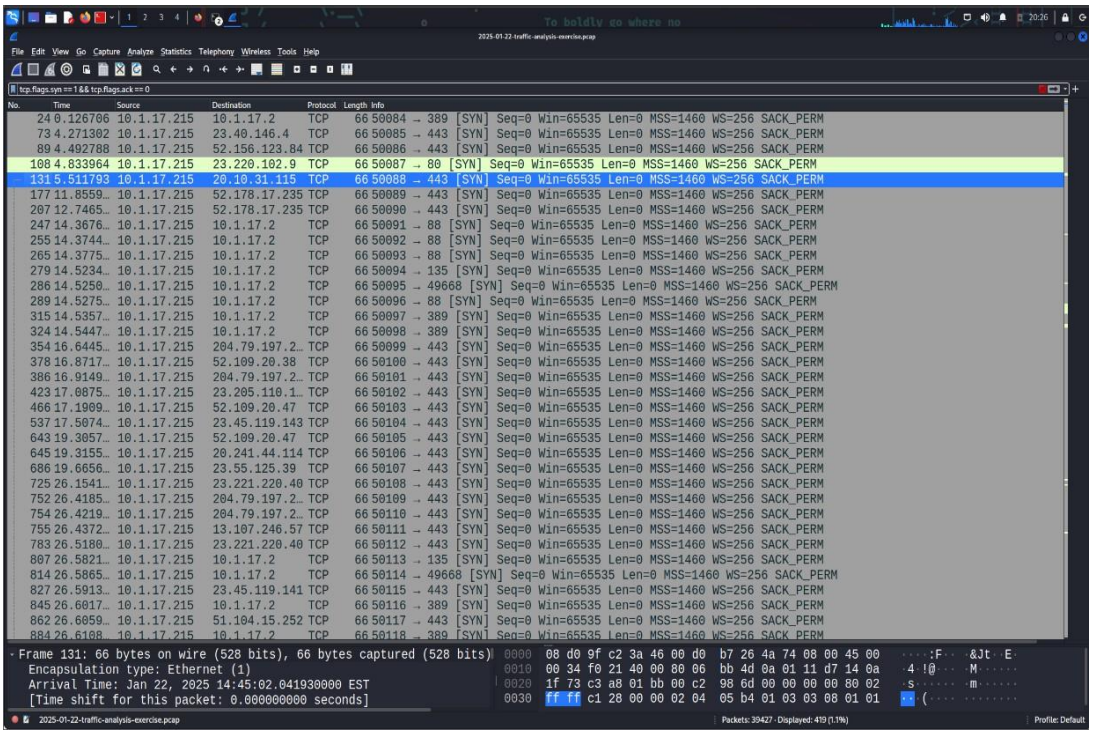


Figure 1: Wireshark capture filtered using tcp.flags.syn==1 && tcp.flags.ack==0, showing several TCP SYN packets from source IP 10.1.17.215 directed at various destination IPs and ports, characteristic of a SYN flood attack seeking to overwhelm the target system.

Packet Details:

- TCP SYN packets sent from source IP 10.1.17.215 (likely spoofed)
- Destination Ports: Primarily **443 (HTTPS)**, with a few on **port 80 (HTTP)**

Observation: The system developed half-open connection queue backlog.

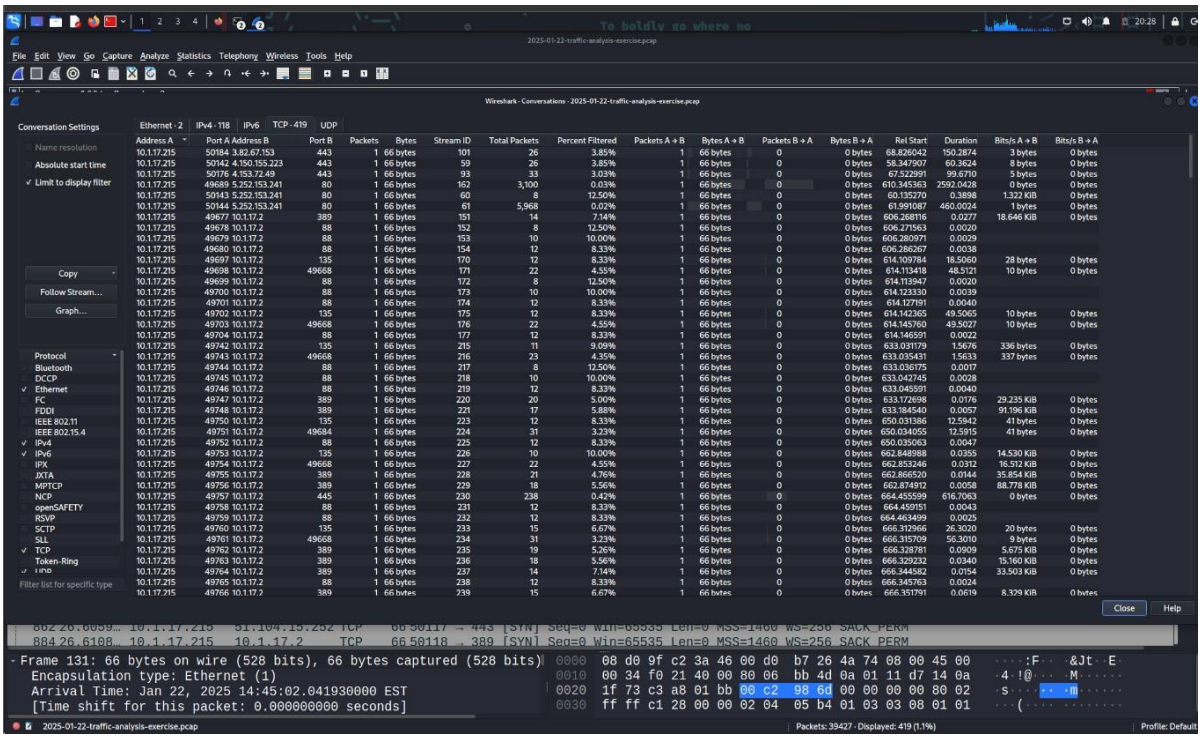


Figure 2: TCP conversation summary of a big mass of one-way SYN packets without responses indicating numerous half-opened connections, classic indication of SYN flood attack attempting to consume server resources..

Mitigation:

- Use SYN cookies.
- Increase the backlog queue size.
- Firewalls with DoS protection.

2.4 Attack 3: UDP Flood with ICMP Backscatter

Description:

This attack sends UDP packets to any random or known ports in order to flood the target. As most of the ports are closed, the target system will respond with ICMP Type 3 Code 3 (Port unreachable), essentially announcing the attack indirectly.

Detection:

Wireshark filter used**:**

- icmp.type == 3 && icmp.code == 3 *(to find ICMP port unreachable)*
- udp && ip.src == 10.1.17.215 && ip.dst == 10.1.17.2 *(for UDP flow)*

Conversation analysis**:**

- 20 UDP packets from source to target, no replies, confirmed ICMP responses.

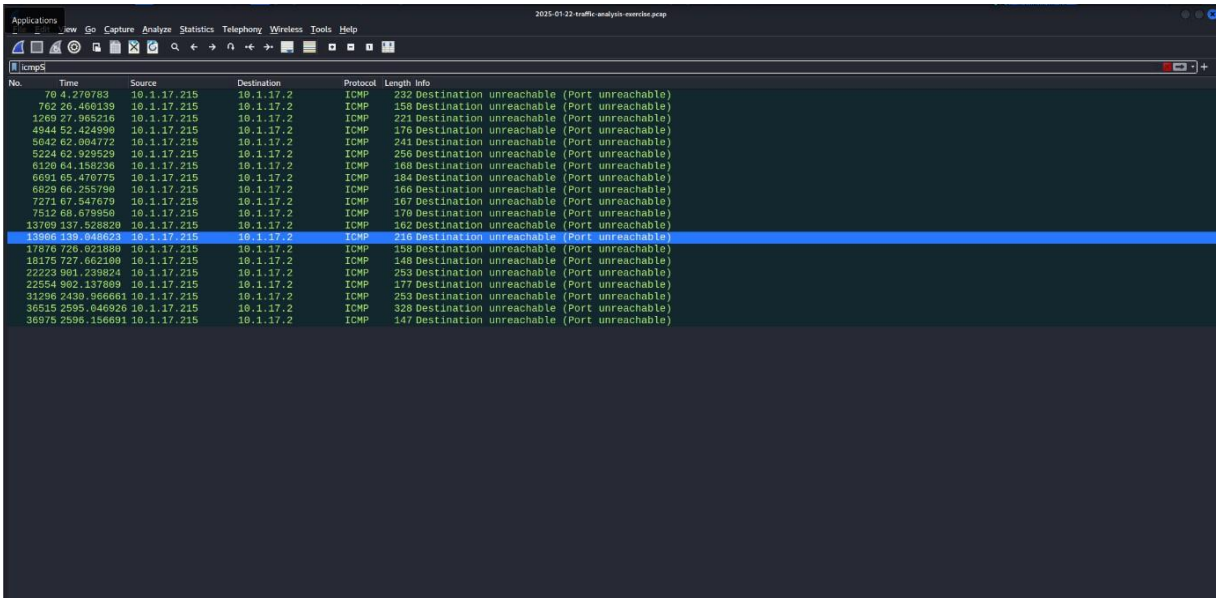


Fig: The screenshot shows ICMP traffic in Wireshark with a series of "Destination unreachable (Port unreachable)" responses between source IP 10.1.17.215 and destination IP 10.1.17.2 and could be evidence of a scan or malformed attempt at communication.

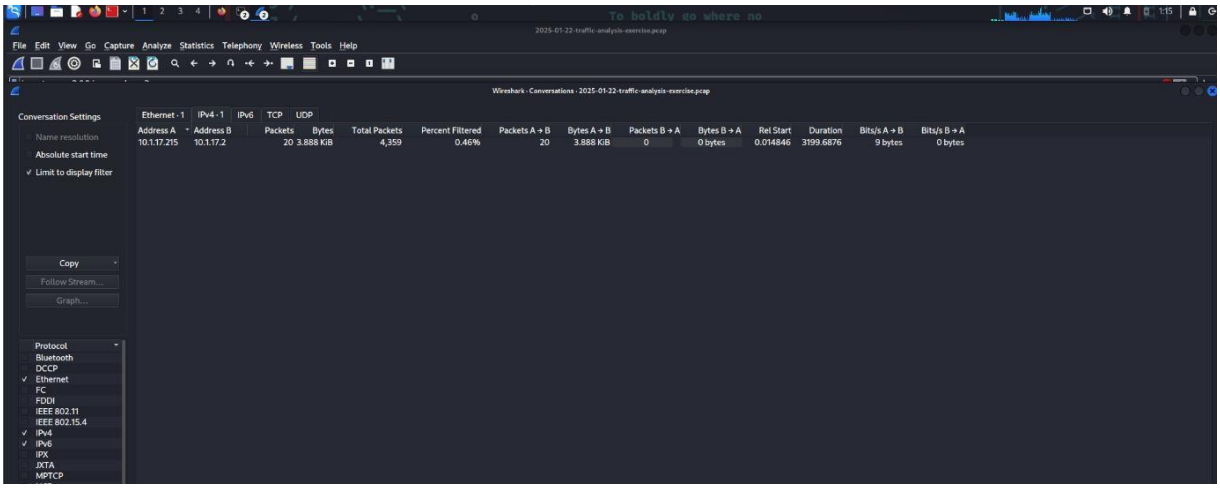


Fig: The screenshot indicates the IPv4 conversation summary, which indicates a one-way communication pattern of 20 packets sent from 10.1.17.215 to 10.1.17.2 and zero replies, which may indicate reconnaissance or unsuccessful UDP-based probing.

Observation: Frequent and large DNS queries to external resolvers were observed.

Mitigation:

- Restrict ICMP rate responses on endpoints .
- Enforce firewalls to recognize and block hostile UDP traffic.
- Detect outbound flood patterns with IDS signatures.

2.5 Attack 4: HTTP-Based Malware or C2

Description:

Malicious HTTP traffic has been identified using Microsoft BITS (User-Agent: Microsoft BITS/7.8) originating from host 10.1.17.215 to msedge.b.tlu.dl.delivery.mp.microsoft.com. This is a typical occurrence of legitimate cover traffic to mask C2 activity via covert file transfers.

Detection:

Wireshark Filters:

- http.request && http.user_agent contains "BITS"
- http && ip.src == 10.1.17.215 && ip.dst == 199.232.214.172
- http.request.method == "HEAD"

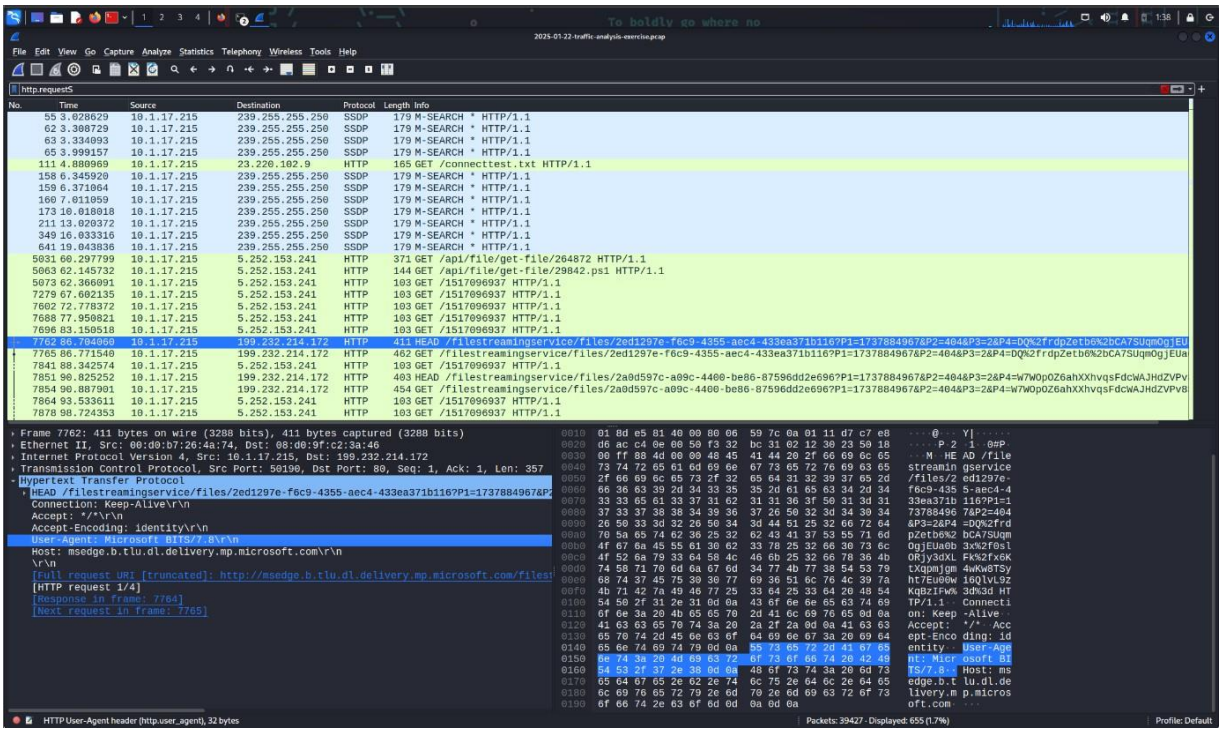


Fig: This Wireshark screen capture demonstrates an HTTP HEAD request from 10.1.17.215 to msedge.b.tlu.dl.delivery.mp.microsoft.com with User-Agent: Microsoft BITS/7.8, commonly used by malware to silently download files. The URL requested has extremely lengthy encoded parameters, which could be a sign of obfuscation.

Key Findings:

- Empowered automated HEAD and GET requests via high obfuscation techniques to formulate query strings.
- The server returns a HTTP 200 OK result for a successful payload delivery.
- The absence of browser activity indicates abuse by a background service.

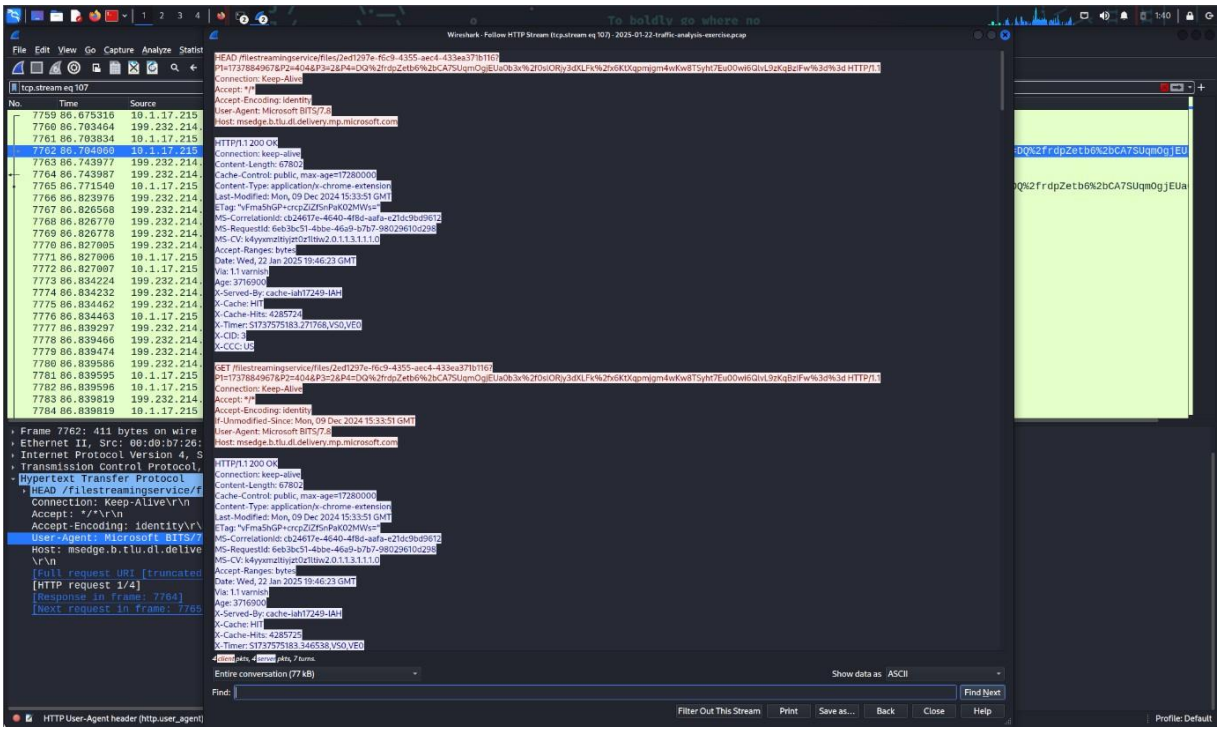


Fig: The below HTTP stream is a successful and accepted response (HTTP 200 OK) to a GET request sent by the same BITS User-Agent. The encoded query string and file path suggest a payload or command retrieval that is typical of advanced persistent threat (APT) techniques.

Mitigation:

- Enable the detection and reporting of suspicious BITS traffic and extra-large query strings.
- If BITS is not being put to productive use, it could be filtered out via GPO.
- Repetitive HEAD/GET patterns can be treated as regular updates, though.

References

These are primarily the articles and nicknamed works referenced in this document. ..... Use MyBib to generate citations:

1. Conti, M., Dragoni, N. and Lesyk, V. (2016). A survey of man in the middle attacks.
2. Kizza, J.M. (2020). Guide to Computer Network Security.
3. Zhou, W. et al. (2018). Distributed denial-of-service attacks and defense mechanisms.
4. Antonakakis, M. et al. (2017). Understanding the Mirai Botnet.
5. Halfond, W.G., Viegas, J. and Orso, A. (2006). A classification of SQL-injection attacks and countermeasures.
6. Bilge, L. and Dumitras, T. (2012). Before we knew it: An empirical study of zero-day attacks in the real world.
7. Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon.
8. Sidorov, D. and Sgandurra, D. (2019). Detecting DNS Tunneling Using Character Frequency Analysis.
9. Cisco Talos (2018). DNSpionage Campaign Targets Middle East.