

Lab: Network Forensics Analysis of a DDoS Attack

MSCS630: Cybercrime Investigation and Digital Evidence

Dr. Mouatez Karbab



September 8, 2026

1 Lab Overview and Scenario

You are a junior cybercrime investigator assigned to the Digital Forensics Unit (DFU). An urgent ticket has been escalated by the Security Operations Center (SOC). A small, but rapidly growing, e-commerce company, "GadgetGalaxy," reported that one of its customer-facing services — a multiplayer game server the company hosts alongside its storefront — became completely unresponsive for approximately one hour earlier today, resulting in significant revenue loss. Their on-site network team managed to capture a short sample of network traffic, roughly twenty-five seconds, while the incident was still in progress, before their firewall rules eventually blocked the anomalous activity.

Your task is to conduct a preliminary analysis of the provided network traffic capture file (.pcap). The goal is to determine the nature of the attack, identify the immediate target, and provide an initial assessment of the evidence. This analysis will form the basis of the official investigation and will guide the next steps, such as coordinating with Internet Service Providers (ISPs). You must approach this task methodically, documenting your steps and findings in a clear, professional report.

2 Learning Objectives

Upon successful completion of this lab, you will be able to:

- Install and utilize Wireshark, a fundamental tool for network forensics.
- Perform initial triage on a large network capture file using statistical analysis tools like the I/O Graph and Protocol Hierarchy.
- Apply specific display filters in Wireshark to isolate malicious traffic associated with a Denial-of-Service attack.

- Identify the key characteristics and digital evidence of a TCP SYN Flood DDoS attack.
- Pinpoint the victim's IP address and the targeted service port based on traffic patterns.
- Explain the concept of IP address spoofing, recognise the packet-level evidence of it, and say why it presents a major challenge for attacker attribution.
- Document technical findings in a structured, professional preliminary analysis report.

Checkpoints. Each section below ends with a **Checkpoint** question. Answer it before moving on — each one is answerable from the evidence file itself, and each shows you both the Wireshark GUI route and the `tshark` command that produce the answer. The same checkpoints appear in the interactive online version of this lab, which tracks your progress as you answer them.

3 Prerequisites: Tools & Data Setup

This lab requires the installation of Wireshark and the download of a specific network traffic capture file. Follow the instructions below carefully.

3.1 Tool Installation: Wireshark

Wireshark is the world's foremost and widely-used network protocol analyzer. It is essential for any network-related investigation.

- **Official Download Page:** <https://www.wireshark.org/download.html>

3.1.1 GUI Installation (Windows/macOS):

1. Navigate to the official Wireshark download page linked above.
2. Download the appropriate installer for your operating system (Windows Installer 64-bit or macOS Intel/Arm 64-bit Disk Image).
3. Run the installer and follow the on-screen prompts.
4. **Critical for Windows Users:** During the installation process, you will be prompted to install **Npcap**. This component is required for live packet capturing. **Ensure the "Install Npcap" checkbox is selected** and proceed with its installation. The default Npcap settings are sufficient.
5. Complete the Wireshark installation.

3.1.2 CLI Installation (Linux - BASH Environment):

These commands are for Debian-based (like Ubuntu) or RHEL-based (like Fedora, CentOS) systems.

1. Open your terminal.
2. Update your package manager's list of available packages:

```

1 # For Debian/Ubuntu
2 sudo apt update
3
4 # For Fedora/RHEL
5 sudo dnf check-update
6

```

3. Install Wireshark using the appropriate command:

```
1 # For Debian/Ubuntu
2 sudo apt install wireshark -y
3
```

During installation, a pop-up will ask: "Should non-superusers be able to capture packets?". Select <Yes> to allow your user account to use Wireshark without running it as root.

```
1 # For Fedora/RHEL
2 sudo dnf install wireshark -y
3
```

4. Add your user account to the `wireshark` group to ensure you have the necessary permissions. (This step is crucial if you selected <No> in the Ubuntu pop-up or for other distributions).

```
1 sudo usermod -aG wireshark $USER
2
```

5. **Log out and log back in** for the group changes to take effect. You can verify your membership with the `groups` command.

Wireshark installs the command-line tools this lab uses alongside the GUI: `capinfos` for file summaries and `tshark` for filtering and statistics. Every step below can be completed either way.

3.2 Data Source: PCAP File

The evidence file for this lab is a packet capture of a real-world spoofed TCP SYN flood, taken from the StopDDoS `packet-captures` collection, a public archive of attack traffic observed in the wild. The target address has been rewritten to a private address by the collectors; the attack traffic itself is untouched.

- **File:** `DrDoS_SYN.pcap` ¹.
- **Download Link:** https://courses.karbab.net/mscs630/DrDoS_SYN.pcap
- **Description:** This file contains traffic from a Denial-of-Service (DoS) SYN flood attack.

Data Preparation and Integrity Verification:

1. Download the file from the link above. It is a direct download and does not require unzipping.
2. Move the downloaded `DrDoS_SYN.pcap` file to a dedicated lab folder (e.g., `~/Lab1-DDoS`).
3. Open a terminal or command prompt, navigate to the folder containing the file, and verify its integrity. Calculating the file's cryptographic hash ensures it was not corrupted during download. This is a critical step in any digital forensics process.

- **BASH Command (Linux, macOS, WSL):**

```
1 shasum -a 256 DrDoS_SYN.pcap
2
```

¹<https://github.com/StopDDoS/packet-captures>

- **Windows Command Prompt:** there is no `shasum` on Windows; use the built-in `certutil` instead.

```
1 certutil -hashfile DrDoS_SYN.pcap SHA256
2
```

- **GUI alternative:** once the file is open in Wireshark, **Statistics > Capture File Properties** displays the SHA256 of the loaded file.
- **Expected SHA256 Hash:**

```
3c5fc7467a0205fd083115926416064d6f256af7705131adeb909f94ef66e7fa
```

4. Compare the hash output by the command with the expected hash value above. If they match, your evidence file is ready for analysis. If they do not match, please re-download the file.

Checkpoint. What is the SHA256 hash of your copy of `DrDoS_SYN.pcap`, and why must a mismatch stop the investigation rather than merely be noted in passing?

How to get it — GUI: **Statistics > Capture File Properties**. *CLI:* `shasum -a 256 DrDoS_SYN.pcap` (or `certutil -hashfile DrDoS_SYN.pcap SHA256` on Windows).

4 Lab Procedure: Step-by-Step Analysis

Follow these steps methodically to analyze the evidence file.

4.1 Step 1: Initial Triage - Gauging the Traffic Volume

Your first action is to get a high-level overview of the traffic. Is it normal traffic, or is there a "smoking gun"? The I/O graph gives a powerful visual indication of traffic volume over time.

1. Open the Evidence File

- **GUI Instructions:** Launch Wireshark. Navigate to **File > Open** and select the `DrDoS_SYN.pcap` file from your lab folder. It may take a few moments to load.
- **CLI Instructions:** (For reference) The `capinfos` command can give a quick summary of a pcap file.

```
1 capinfos DrDoS_SYN.pcap
2
```

- *Observation/Expected Outcome:* The capture holds roughly 37,800 packets in just under 24 seconds — an average of about 1,600 packets per second sustained across the whole file. Note the average packet size as well: 60 bytes. Sixty bytes is a bare Ethernet, IP and TCP header with nothing after it, so not one of these packets is carrying data. Traffic that is all envelope and no letter is the first hint that you are looking at a flood rather than at customers.

2. Visualize Traffic with the I/O Graph

- **GUI Instructions:** In Wireshark, navigate to **Statistics > I/O Graph**. A new window will open. By default, it shows packets per second.
- **CLI Instructions:** `tshark` prints the same per-second counts that the graph draws.

```
1 tshark -r DrDoS_SYN.pcap -q -z io,stat,1
2
```

- *Observation/Expected Outcome:* The traffic arrives in volleys rather than as a steady stream. The first second alone carries over 23,000 packets; a second burst of around 10,000 packets per second follows at about the three-second mark; several seconds of near silence come after it; and the last ten seconds settle into a low, steady tail of roughly 60 to 95 packets per second. Record the peak rate — it is the quantitative measure of impact your report needs. Note that this capture contains no legitimate baseline traffic to compare against: the sample was taken while the incident was already under way, so everything on the graph is attack.

Checkpoint. How many packets does the capture contain, over what duration, and what is the average packet size? What does that average size tell you about the contents of these packets?

How to get it — GUI: Statistics > Capture File Properties.

CLI:

```
1 capinfos DrDoS_SYN.pcap
2 tshark -r DrDoS_SYN.pcap -Y "tcp.len > 0" | wc -l
```

4.2 Step 2: Identifying the Dominant Protocol

Now that you’ve confirmed a traffic flood, you need to determine what *kind* of traffic it is. The Protocol Hierarchy statistic breaks down the traffic by protocol.

1. Analyze the Protocol Hierarchy

- **GUI Instructions:** In the main Wireshark window, navigate to Statistics > Protocol Hierarchy.
- **CLI Instructions:** The `tshark` command-line utility (installed with Wireshark) can generate this information directly.

```
1 tshark -r DrDoS_SYN.pcap -q -z io,phs
2
```

- *Observation/Expected Outcome:* Every frame in the capture is **Transmission Control Protocol (TCP)** — 100% of the packets and 100% of the bytes. This narrows your focus significantly; the attack is TCP-based. Pay attention to what is *missing* as well: there is no UDP, no ICMP, and nothing at the application layer at all — no HTTP, no TLS, no DNS.

Checkpoint. What percentage of the capture is TCP, and which protocols that you would expect on a busy production server are completely absent? What does that absence tell you about the capture?

How to get it — GUI: Statistics > Protocol Hierarchy.

CLI:

```
1 tshark -r DrDoS_SYN.pcap -q -z io,phs
```

4.3 Step 3: Isolating the Attack Signature

Based on the scenario and the protocol dominance, a SYN flood is a primary suspect. A SYN flood attack consists of a high volume of TCP SYN packets (the first step in a connection) sent without completing the handshake. You will now use a display filter to isolate only these packets.

1. Apply the SYN Flag Filter

- **GUI Instructions:** In the main Wireshark window, locate the green "Apply a display filter" bar at the top. Type the following filter and press Enter:

```
tcp.flags.syn == 1 and tcp.flags.ack == 0
```

This filter shows only packets where the SYN flag is set (`== 1`) but the ACK flag is not set (`== 0`), which uniquely identifies the initial connection request packet in a TCP handshake.

- **CLI Instructions:** You can achieve the same filtering with `tshark`. The `-Y` flag is for a display filter.

```
1 # This command will print a summary of the first 10 matching packets
2 tshark -r DrDoS_SYN.pcap -Y "tcp.flags.syn == 1 and tcp.flags.ack == 0
   " | head -n 10
3
```

- *Observation/Expected Outcome:* Nothing appears to change in the packet list — and that is the finding. Read the status bar at the bottom of the Wireshark window: it reports **Displayed: 37841 (100.0%)**. Every single packet in the capture is a bare connection request. Now change the filter to `tcp.flags.syn == 1 and tcp.flags.ack == 1` to look for the server's replies. There are none. A busy server under legitimate load produces SYN, SYN-ACK and ACK in roughly equal measure; a capture of nothing but unanswered SYNs is a **SYN Flood**.

Checkpoint. How many packets match the SYN filter, and how many completed handshakes (SYN-ACK replies) does the capture contain?

How to get it — GUI: apply each filter in turn and read **Displayed:** in the status bar.

CLI:

```
1 tshark -r DrDoS_SYN.pcap -Y "tcp.flags.syn==1 and tcp.flags.ack==0" | wc -l
2 tshark -r DrDoS_SYN.pcap -Y "tcp.flags.syn==1 and tcp.flags.ack==1" | wc -l
```

4.4 Step 4: Pinpointing the Victim, the Target Service and the Source

With the attack traffic isolated, you can now identify the target and analyze the source of the attack.

1. Identify the Victim (Destination IP)

- **GUI Instructions:** Open **Statistics > Endpoints** and select the IPv4 tab, then sort by the **Packets** column. Do not scroll the packet list looking for a pattern — with 37,841 packets that does not scale, and the Endpoints view answers the question in one screen.

- **CLI Instructions:**

```
1 tshark -r DrDoS_SYN.pcap -q -z endpoints,ip | head
2
```

- *Observation/Expected Outcome:* A single address sits far above every other, receiving every packet in the capture. That is the victim server. Look at its **Tx** and **Rx** columns as well: it received all 37,841 packets and transmitted nothing at all. Record this IP address for your report.

2. Identify the Targeted Service (Destination Port)

- **GUI Instructions:** With the SYN filter still applied, read the destination port in the Info column, or open **Statistics > Conversations** and select the TCP tab.
- **CLI Instructions:**

```
1 tshark -r DrDoS_SYN.pcap -T fields -e tcp.dstport | sort | uniq -c |
  sort -nr | head
2
```

- *Observation/Expected Outcome:* Every packet is aimed at one destination port. The flood is not directed at the machine in general but at one specific service running on it, which tells you which of GadgetGalaxy's services was taken down. It is also the detail a responder needs in order to scope a mitigation rule: a filter narrowed to one port protects the rest of the host.

3. Analyze the Attacker(s) (Source IP)

- **GUI Instructions:** Look at the "Source" column in the packet list and scroll through it, then compare against the IPv4 tab of **Statistics > Endpoints**.
- **CLI Instructions:** A powerful way to see the distribution of source IPs is to have tshark extract them, count the unique occurrences, and sort them.

```
1 # Extracts, counts, and sorts the source IPs by frequency
2 tshark -r DrDoS_SYN.pcap -Y "tcp.flags.syn==1 and tcp.flags.ack==0" -T
  fields -e ip.src | sort | uniq -c | sort -nr | head -n 20
3
4 # Counts how many distinct source addresses appear in total
5 tshark -r DrDoS_SYN.pcap -Y "tcp.flags.syn==1 and tcp.flags.ack==0" -T
  fields -e ip.src | sort -u | wc -l
6
```

This command does the following:

- **-T fields -e ip.src:** Extracts only the source IP address field.
- **sort | uniq -c:** Counts the unique occurrences of each IP.
- **sort -nr:** Sorts the result numerically in reverse order (most frequent first).
- **head -n 20:** Shows the top 20 most frequent source IPs.
- *Observation/Expected Outcome:* Unlike the single destination IP, the source addresses are almost all different: 37,623 distinct addresses across 37,841 packets, with even the busiest source appearing only twice. They span 216 different /8 prefixes, which is close to the entire routable IPv4 address space. That is not the shape of a botnet, where a few thousand compromised machines each send many packets. It is the shape of a **spoofed source address**, forged afresh on nearly every packet so the attacker's real IP never appears.

4. Prove the Spoofing (Time to Live)

- **GUI Instructions:** Select any packet, expand the **Internet Protocol Version 4** tree in the detail pane, right-click the **Time to Live** field and choose **Apply as Column**. Sort the new column.
- **CLI Instructions:**

```
1 tshark -r DrDoS_SYN.pcap -T fields -e ip.ttl | sort | uniq -c
2
```

- *Observation/Expected Outcome:* A large source count is suggestive, but on its own it is not proof — a genuinely large botnet would also show many addresses. The TTL field settles it. Every IP packet leaves its sender with an operating-system default

TTL and loses one for each router it crosses, so packets arriving from thousands of genuinely different hosts across the internet arrive with a wide spread of TTL values. Here there are only two distinct values in the entire capture. Tens of thousands of supposedly independent senders scattered across the whole address space cannot all be exactly the same number of hops away. They are one sender, on one or two paths, forging the source field.

Checkpoint. Give the victim's IP address, the targeted destination port, and the number of distinct source addresses. Then: how many distinct TTL values appear in the capture, and why does that number contradict the claim that the traffic came from tens of thousands of separate hosts?

How to get it — GUI: Statistics > Endpoints (IPv4 tab) for the victim; Statistics > Conversations (TCP tab) for the port; Apply as Column on the Time to Live field for the TTL.

CLI:

```
1 tshark -r DrDoS_SYN.pcap -q -z endpoints,ip | head
2 tshark -r DrDoS_SYN.pcap -T fields -e tcp.dstport | sort -u
3 tshark -r DrDoS_SYN.pcap -T fields -e ip.src | sort -u | wc -l
4 tshark -r DrDoS_SYN.pcap -T fields -e ip.ttl | sort | uniq -c
```

5 Lab Report

Complete the following report template based on your analysis. This document simulates a preliminary report you would submit to your supervisor in the DFU. Write it in your own words: every figure you need is one you produced in Steps 1 to 4, and a finding stated without the evidence that supports it is not a finding.

Cybercrime Investigation - Preliminary Analysis Report

- **Case ID:** *LAB-01-DDoS-[Your-Name]*
- **Investigator:** *[Your Name]*
- **Date of Analysis:** *[Today's Date]*

1. Evidence Details

- **Filename:** *DrDoS_SYN.pcap*
- **SHA256 Hash:** *[Paste the SHA256 hash value you calculated during data setup]*
- **Capture Window:** *[Start time, end time and duration, from Capture File Properties]*

2. Executive Summary

[Three or four sentences, written for a supervisor who will not read the rest of the report: what happened, to whom, by what mechanism, and why identifying the attacker is not straightforward. No packet-level detail here — that belongs in Section 3.]

3. Key Findings

- **Attack Type Identified:** *[Name the attack type, then give the evidence that establishes it. Two observations are enough if you choose them well.]*
- **Victim IP Address:** *[From Step 4.]*
- **Targeted Service Port:** *[From Step 4. State why a single destination port matters to whoever has to write the mitigation rule.]*
- **Attack Source:** *[Characterise the sources: how many distinct addresses, how many packets each, how widely distributed. Then say what you conclude from that, and note that this evidence alone is suggestive rather than conclusive.]*
- **Key Indicators:**
 - **Quantitative:** *[Peak rate from the I/O graph: _____ packets/sec. Total packets: _____. Capture duration: _____. TCP share of the capture: _____. Distinct source addresses: _____. Completed handshakes: _____.]*
 - **Qualitative:** *[In one or two sentences, describe the packet-level signature: which flags are set, what the destination fields do across the capture, what the source fields do, and what the server's response was.]*
 - **Proof of Spoofing:** *[State the number of distinct TTL values you found and explain, in one sentence, why that number rules out a genuinely distributed set of senders.]*

4. Attribution Challenges

[Given what you established about the source addresses, what can and cannot be concluded about who carried out this attack? What would a next investigative step look like, and who would you need to involve? Be specific about the limits of this evidence.]
