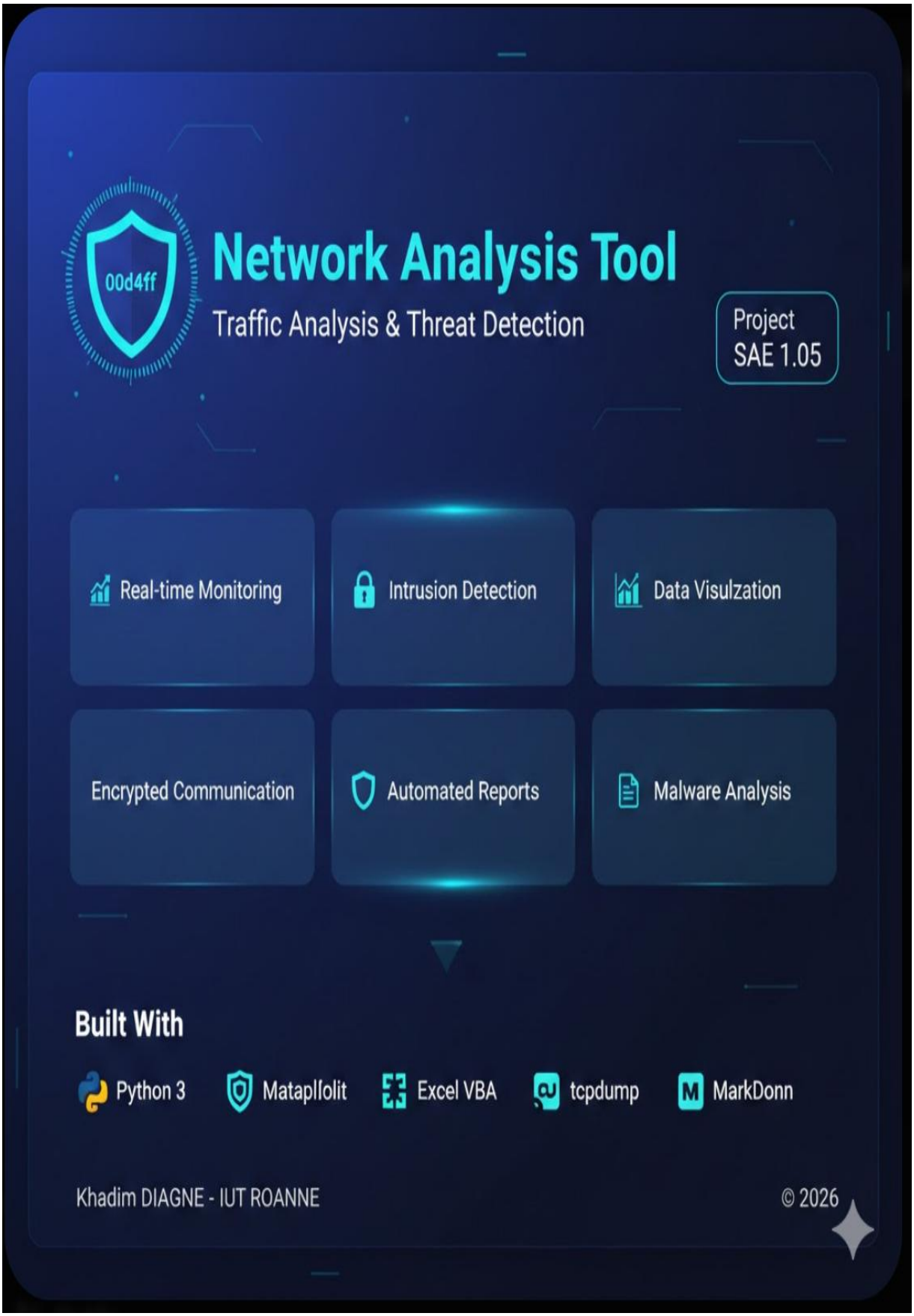

00d4ff
Network Analysis Tool
Traffic Analysis & Threat Detection
Project
SAE 1.05
Real-time Monitoring
Intrusion Detection
Data Visulzation
Encrypted Communication
Automated Reports
Malware Analysis
Built With
Python 3
Mataplfolit
Excel VBA
tcpdump
MarkDonn
Khadim DIAGNE - IUT ROANNE
© 2026

Table of Contents

I. What is this about?

We had a network performance issue at the India site: very slow connections and a lot of packet loss.
To understand what was going on, we needed to analyze tcpdump captures and identify the root cause.

To make this easier, I created a small toolkit with:

- **2 Python scripts**
- **1 Excel macro**

Once you understand the logic, it's quite simple to use.

Overview of the toolkit:

- **Script 1:** Converts raw tcpdump files into a clean CSV file
- **Script 2:** Analyzes the CSV file and generates clear charts
- **Excel macro:** Provides additional, optional analysis

We'll go through everything step by step.

II. Setup (do this first)

You need **Python 3** installed.

If you don't have it yet:

- Go to **python.org** and download Python 3
- During installation, make sure to **check the box "Add Python to PATH"** (this is important)

Then open a **command prompt or terminal** and run:

pip install matplotlib

This installs **matplotlib**, the library used to create the charts.
The installation usually takes less than a minute.

This document provides instructions on how to use our Python toolset to process raw network captures and identify security threats.

What this project does

This tool helps you find out why a network is having problems. It reads network traffic files (tcpdump format) and detects suspicious activities such as SSH brute force attacks, port scans, and ICMP floods.

This project was developed for a university assignment at IUT Roanne: two company sites (France and India) experienced network issues and needed deeper traffic analysis than standard tools provided.

📦 **What's inside**

Scripts and purpose

Script	Purpose
txt_to_csv.py	Converts raw tcpdump text file → clean CSV (semicolon-separated)
csv_to_md.py	Analyzes CSV → generates Markdown report with security alerts
md_to_html.py	Converts Markdown report → styled HTML page for presentation

Files generated

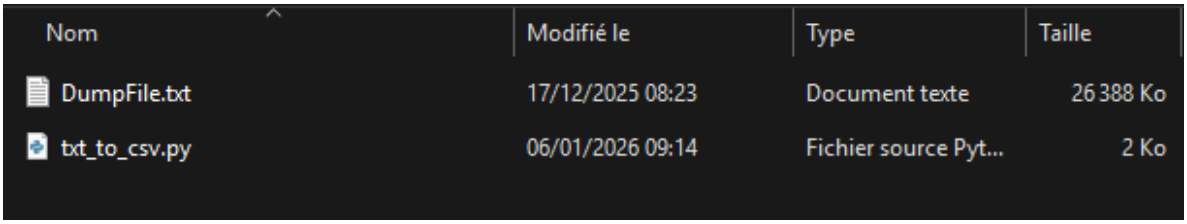
- Network_Analysis.csv — Structured data (semicolon-separated)
- Network_Report.md — Security analysis report (Markdown)
- Network_Report.html — Final styled report (HTML with embedded CSS)

III. Phase 1: Data Structuring (TXT to CSV)

Before starting the analysis, we must transform the raw tcpdump output into a structured format.

- **Setup:** Place both DumpFile.txt and txt_to_csv.py in the same folder.
- **Execution:** Run the script using python txt_to_csv.py.
- **Result:** The script parses the text and generates **Network_Analysis.csv**.

Note: This step is crucial because it extracts timestamps, source/destination IPs, and technical flags into a table format that is easy to manipulate.

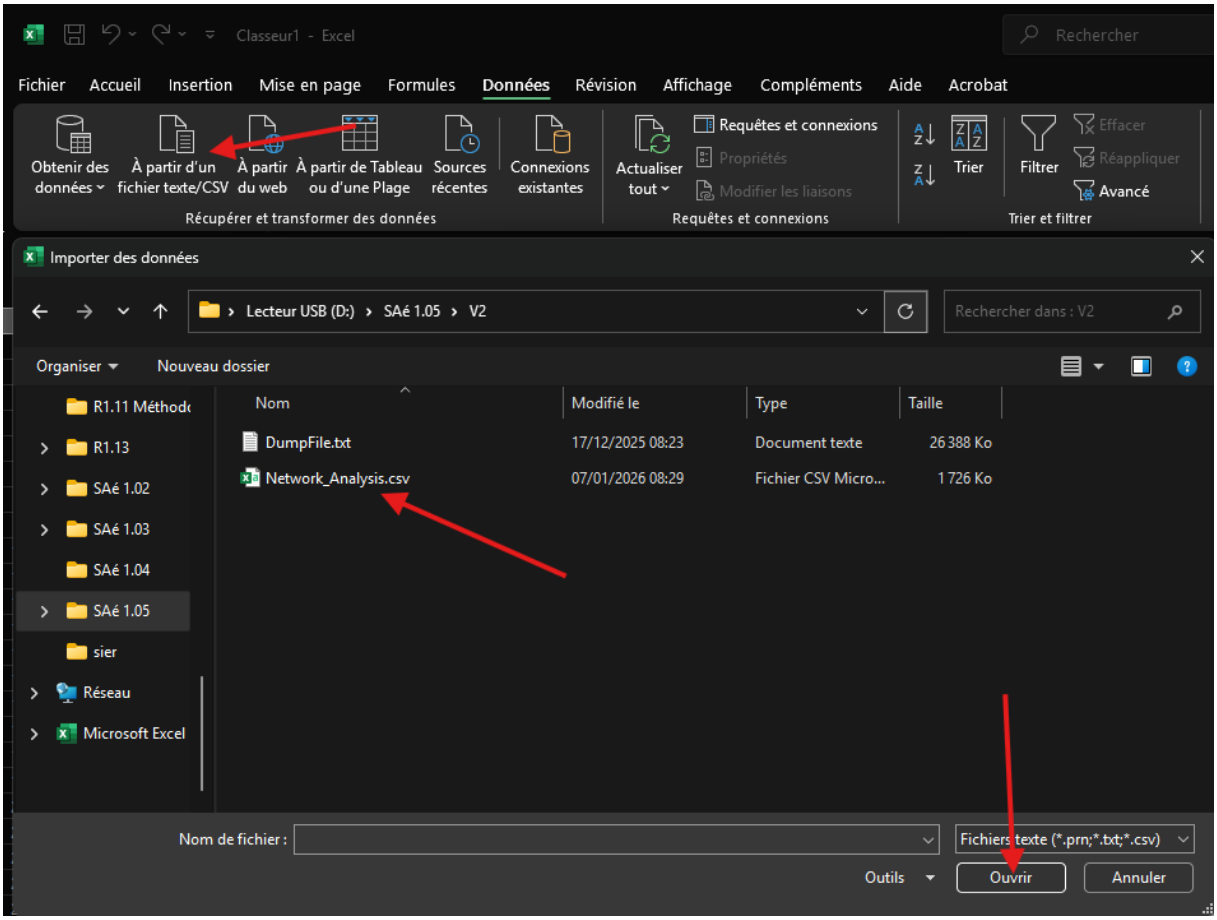


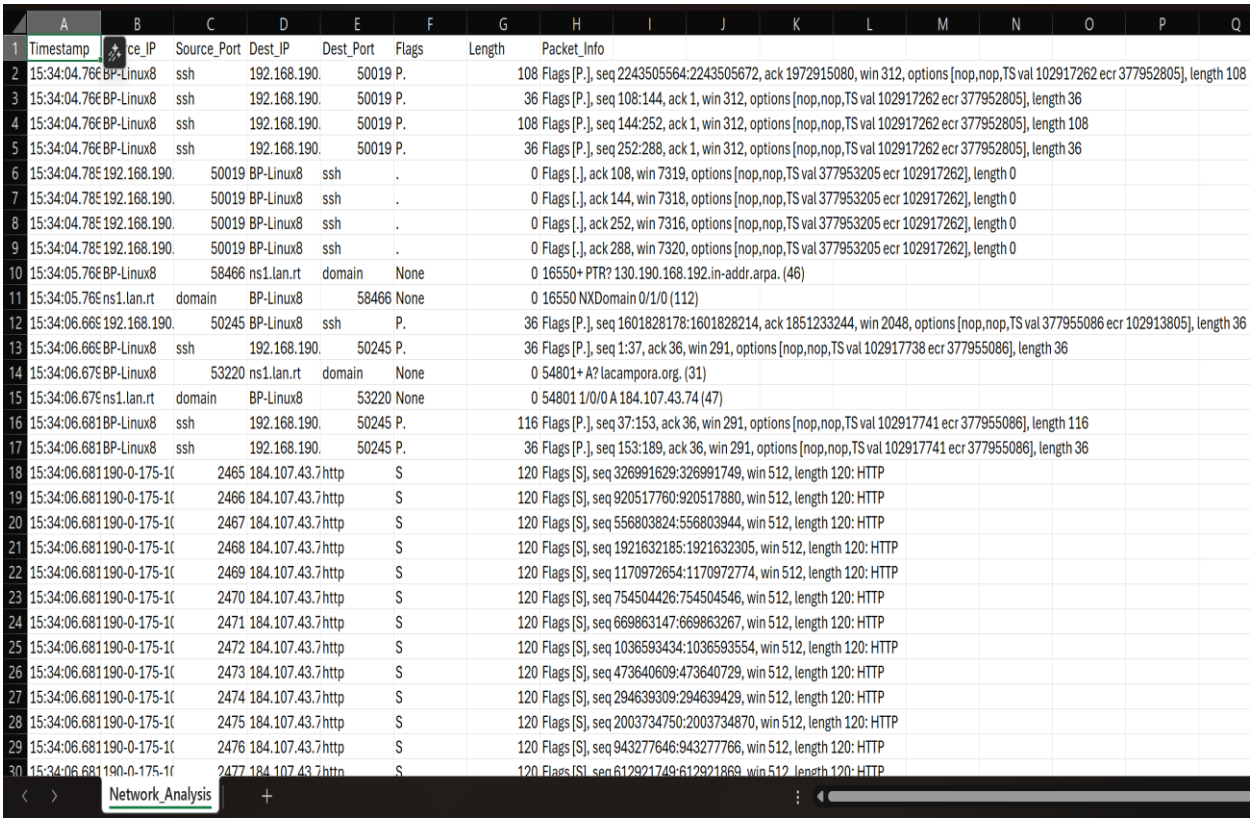
IV. Phase 2: Manual Analysis in Excel

Once you have the CSV file, you can perform a deep dive into the data using Microsoft Excel to visualize the attack patterns.

Steps for Excel Analysis:

1. **Import:** Open Excel and import Network_Analysis.csv (Data > From Text/CSV). Ensure the delimiter is set to **Semicolon ;**
2.

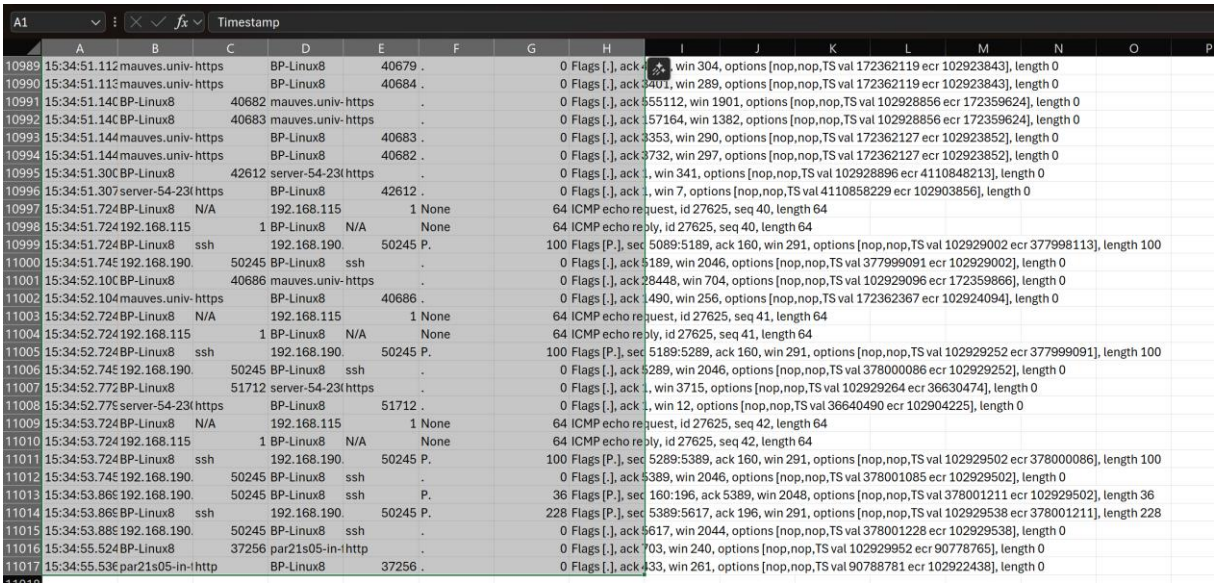




	A	B	C	D	E	F	G	H
1	Timestamp	ce_IP	Source_Port	Dest_IP	Dest_Port	Flags	Length	Packet_Info
2	15:34:04.766	BP-Linux8	ssh	192.168.190.	50019	P.	108	Flags [P.], seq 2243505564:2243505672, ack 1972915080, win 312, options [nop,nop,TS val 102917262 ecr 377952805], length 108
3	15:34:04.766	BP-Linux8	ssh	192.168.190.	50019	P.	36	Flags [P.], seq 108:144, ack 1, win 312, options [nop,nop,TS val 102917262 ecr 377952805], length 36
4	15:34:04.766	BP-Linux8	ssh	192.168.190.	50019	P.	108	Flags [P.], seq 144:252, ack 1, win 312, options [nop,nop,TS val 102917262 ecr 377952805], length 108
5	15:34:04.766	BP-Linux8	ssh	192.168.190.	50019	P.	36	Flags [P.], seq 252:288, ack 1, win 312, options [nop,nop,TS val 102917262 ecr 377952805], length 36
6	15:34:04.785	192.168.190.	50019	BP-Linux8	ssh	.	0	Flags [.], ack 108, win 7319, options [nop,nop,TS val 377953205 ecr 102917262], length 0
7	15:34:04.785	192.168.190.	50019	BP-Linux8	ssh	.	0	Flags [.], ack 144, win 7318, options [nop,nop,TS val 377953205 ecr 102917262], length 0
8	15:34:04.785	192.168.190.	50019	BP-Linux8	ssh	.	0	Flags [.], ack 252, win 7316, options [nop,nop,TS val 377953205 ecr 102917262], length 0
9	15:34:04.785	192.168.190.	50019	BP-Linux8	ssh	.	0	Flags [.], ack 288, win 7320, options [nop,nop,TS val 377953205 ecr 102917262], length 0
10	15:34:05.766	BP-Linux8	58466	ns1.lan.rt	domain	None	0	16550+ PTR? 130.190.168.192.in-addr.arpa. (46)
11	15:34:05.766	ns1.lan.rt	domain	BP-Linux8	58466	None	0	16550 NXDomain 0/1/0 (112)
12	15:34:06.666	192.168.190.	50245	BP-Linux8	ssh	P.	36	Flags [P.], seq 1601828178:1601828214, ack 1851233244, win 2048, options [nop,nop,TS val 377955086 ecr 102913805], length 36
13	15:34:06.666	BP-Linux8	ssh	192.168.190.	50245	P.	36	Flags [P.], seq 1:37, ack 36, win 291, options [nop,nop,TS val 102917738 ecr 377955086], length 36
14	15:34:06.676	BP-Linux8	53220	ns1.lan.rt	domain	None	0	54801+ A? lacampora.org. (31)
15	15:34:06.676	ns1.lan.rt	domain	BP-Linux8	53220	None	0	54801 1/0/0 A 184.107.43.74 (47)
16	15:34:06.681	BP-Linux8	ssh	192.168.190.	50245	P.	116	Flags [P.], seq 37:153, ack 36, win 291, options [nop,nop,TS val 102917741 ecr 377955086], length 116
17	15:34:06.681	BP-Linux8	ssh	192.168.190.	50245	P.	36	Flags [P.], seq 153:189, ack 36, win 291, options [nop,nop,TS val 102917741 ecr 377955086], length 36
18	15:34:06.681	190-0-175-1(	2465	184.107.43.7	http	S	120	Flags [S], seq 326991629:326991749, win 512, length 120: HTTP
19	15:34:06.681	190-0-175-1(	2466	184.107.43.7	http	S	120	Flags [S], seq 920517760:920517880, win 512, length 120: HTTP
20	15:34:06.681	190-0-175-1(	2467	184.107.43.7	http	S	120	Flags [S], seq 556803824:556803944, win 512, length 120: HTTP
21	15:34:06.681	190-0-175-1(	2468	184.107.43.7	http	S	120	Flags [S], seq 1921632185:1921632305, win 512, length 120: HTTP
22	15:34:06.681	190-0-175-1(	2469	184.107.43.7	http	S	120	Flags [S], seq 1170972654:1170972774, win 512, length 120: HTTP
23	15:34:06.681	190-0-175-1(	2470	184.107.43.7	http	S	120	Flags [S], seq 754504426:754504546, win 512, length 120: HTTP
24	15:34:06.681	190-0-175-1(	2471	184.107.43.7	http	S	120	Flags [S], seq 669863147:669863267, win 512, length 120: HTTP
25	15:34:06.681	190-0-175-1(	2472	184.107.43.7	http	S	120	Flags [S], seq 1036593434:1036593554, win 512, length 120: HTTP
26	15:34:06.681	190-0-175-1(	2473	184.107.43.7	http	S	120	Flags [S], seq 473640609:473640729, win 512, length 120: HTTP
27	15:34:06.681	190-0-175-1(	2474	184.107.43.7	http	S	120	Flags [S], seq 294639309:294639429, win 512, length 120: HTTP
28	15:34:06.681	190-0-175-1(	2475	184.107.43.7	http	S	120	Flags [S], seq 2003734750:2003734870, win 512, length 120: HTTP
29	15:34:06.681	190-0-175-1(	2476	184.107.43.7	http	S	120	Flags [S], seq 943277646:943277766, win 512, length 120: HTTP
30	15:34:06.681	190-0-175-1(	2477	184.107.43.7	http	S	120	Flags [S], seq 612921749:612921869, win 512, length 120: HTTP

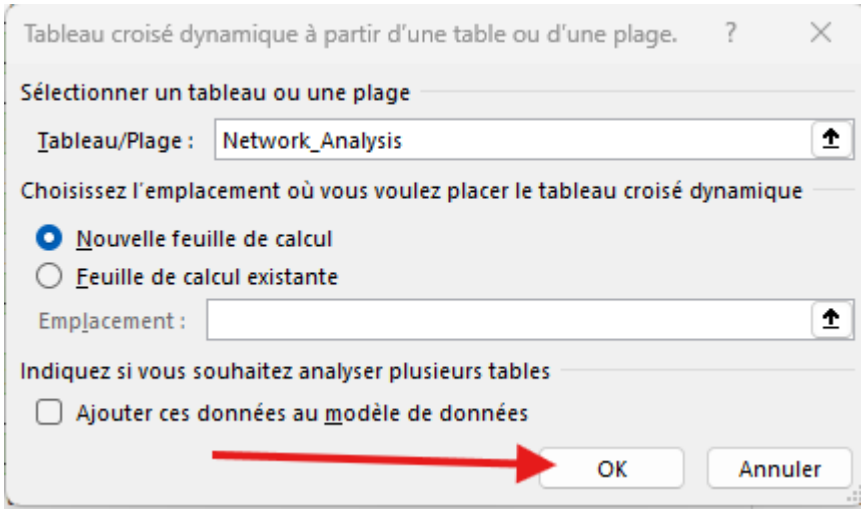
Network_Analysis

3. **Select Data:** Click any cell in your data and press **Ctrl+A** to select the entire table.
4. **Create Pivot Table:** Go to the **Insert** tab and click on **PivotTable**. Click **OK** to create it in a new worksheet.



A1 Timestamp

	A	B	C	D	E	F	G	H
10989	15:34:51.112	mauves.univ-	https	BP-Linux8	40679	.	0	Flags [.], ack win 304, options [nop,nop,TS val 172362119 ecr 102923843], length 0
10990	15:34:51.113	mauves.univ-	https	BP-Linux8	40684	.	0	Flags [.], ack win 289, options [nop,nop,TS val 172362119 ecr 102923843], length 0
10991	15:34:51.140	BP-Linux8	40682	mauves.univ-	https	.	0	Flags [.], ack win 1901, options [nop,nop,TS val 102928856 ecr 172359624], length 0
10992	15:34:51.140	BP-Linux8	40683	mauves.univ-	https	.	0	Flags [.], ack win 1382, options [nop,nop,TS val 102928856 ecr 172359624], length 0
10993	15:34:51.144	mauves.univ-	https	BP-Linux8	40683	.	0	Flags [.], ack win 290, options [nop,nop,TS val 172362127 ecr 102923852], length 0
10994	15:34:51.144	mauves.univ-	https	BP-Linux8	40682	.	0	Flags [.], ack win 297, options [nop,nop,TS val 172362127 ecr 102923852], length 0
10995	15:34:51.307	BP-Linux8	42612	server-54-23(	https	.	0	Flags [.], ack win 341, options [nop,nop,TS val 102928896 ecr 4110848213], length 0
10996	15:34:51.307	server-54-23(	https	BP-Linux8	42612	.	0	Flags [.], ack win 7, options [nop,nop,TS val 4110858229 ecr 102903856], length 0
10997	15:34:51.724	BP-Linux8	N/A	192.168.115	1	None	64	ICMP echo request, id 27625, seq 40, length 64
10998	15:34:51.724	192.168.115	1	BP-Linux8	N/A	None	64	ICMP echo reply, id 27625, seq 40, length 64
10999	15:34:51.724	BP-Linux8	ssh	192.168.190.	50245	P.	100	Flags [P.], seq 5089:5189, ack 160, win 291, options [nop,nop,TS val 102929002 ecr 377998113], length 100
11000	15:34:51.745	192.168.190.	50245	BP-Linux8	ssh	.	0	Flags [.], ack 5189, win 2046, options [nop,nop,TS val 377999091 ecr 102929002], length 0
11001	15:34:52.100	BP-Linux8	40686	mauves.univ-	https	.	0	Flags [.], ack 28448, win 704, options [nop,nop,TS val 102929096 ecr 172359866], length 0
11002	15:34:52.104	mauves.univ-	https	BP-Linux8	40686	.	0	Flags [.], ack 1490, win 256, options [nop,nop,TS val 172362367 ecr 102924094], length 0
11003	15:34:52.724	BP-Linux8	N/A	192.168.115	1	None	64	ICMP echo request, id 27625, seq 41, length 64
11004	15:34:52.724	192.168.115	1	BP-Linux8	N/A	None	64	ICMP echo reply, id 27625, seq 41, length 64
11005	15:34:52.724	BP-Linux8	ssh	192.168.190.	50245	P.	100	Flags [P.], seq 5189:5289, ack 160, win 291, options [nop,nop,TS val 102929252 ecr 377999091], length 100
11006	15:34:52.745	192.168.190.	50245	BP-Linux8	ssh	.	0	Flags [.], ack 5289, win 2046, options [nop,nop,TS val 378000086 ecr 102929252], length 0
11007	15:34:52.772	BP-Linux8	51712	server-54-23(	https	.	0	Flags [.], ack win 3715, options [nop,nop,TS val 102929264 ecr 36630474], length 0
11008	15:34:52.779	server-54-23(	https	BP-Linux8	51712	.	0	Flags [.], ack win 12, options [nop,nop,TS val 36640490 ecr 102904225], length 0
11009	15:34:53.724	BP-Linux8	N/A	192.168.115	1	None	64	ICMP echo request, id 27625, seq 42, length 64
11010	15:34:53.724	192.168.115	1	BP-Linux8	N/A	None	64	ICMP echo reply, id 27625, seq 42, length 64
11011	15:34:53.724	BP-Linux8	ssh	192.168.190.	50245	P.	100	Flags [P.], seq 5289:5389, ack 160, win 291, options [nop,nop,TS val 102929502 ecr 378000086], length 100
11012	15:34:53.745	192.168.190.	50245	BP-Linux8	ssh	.	0	Flags [.], ack 5389, win 2046, options [nop,nop,TS val 378001085 ecr 102929502], length 0
11013	15:34:53.866	192.168.190.	50245	BP-Linux8	ssh	P.	36	Flags [P.], seq 160:196, ack 5389, win 2048, options [nop,nop,TS val 378001211 ecr 102929502], length 36
11014	15:34:53.866	BP-Linux8	ssh	192.168.190.	50245	P.	228	Flags [P.], seq 5389:5617, ack 196, win 291, options [nop,nop,TS val 102929538 ecr 378001211], length 228
11015	15:34:53.888	192.168.190.	50245	BP-Linux8	ssh	.	0	Flags [.], ack 5617, win 2044, options [nop,nop,TS val 378001228 ecr 102929538], length 0
11016	15:34:55.524	BP-Linux8	37256	par21s05-in-l	http	.	0	Flags [.], ack 703, win 240, options [nop,nop,TS val 102929952 ecr 90778765], length 0
11017	15:34:55.536	par21s05-in-l	http	BP-Linux8	37256	.	0	Flags [.], ack 433, win 261, options [nop,nop,TS val 90788781 ecr 102922438], length 0

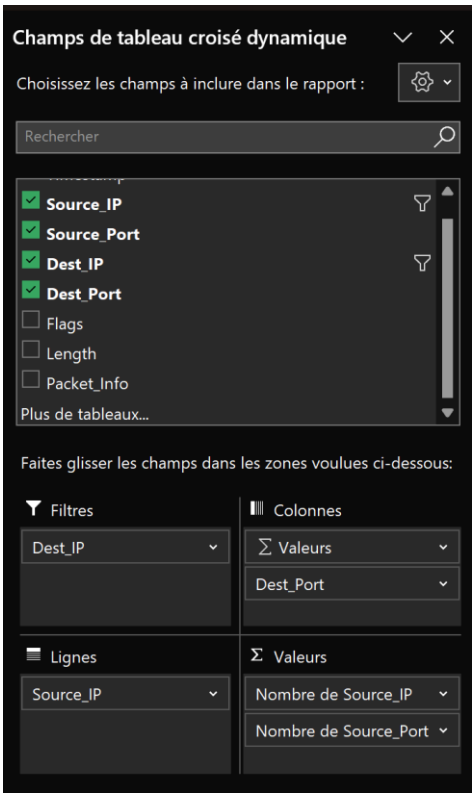


5. **Configure Fields:**

   Drag **Source** into the **Rows** area.

   Drag **Source** again into the **Values** area (it should show "Count of Source").

   Drag **Destination** into the **Filters** are



6. **Identify the Threat:**

Use the Destination filter at the top of the sheet to select only ssh.

	A	B	C	D	E
1	Dest_IP	BP-Linux8			
2					
3		Étiquettes de colonnes			
4		Nombre de Source_IP	Nombre de Source_Port	Total Nombre de Source_IP	Total Nombre de Source_Port
5	Étiquettes de lignes	ssh	ssh		
6	192.168.190.130	66	66	66	66
7	**Total général**	**66**	**66**	**66**	**66**
8					
9					

Right-click on any number in the "Count" column and select Sort > Sort Largest to Smallest.

Finding: You will see IP 192.168.190.130 appearing at the top with different source ports, confirming the Brute Force attack.

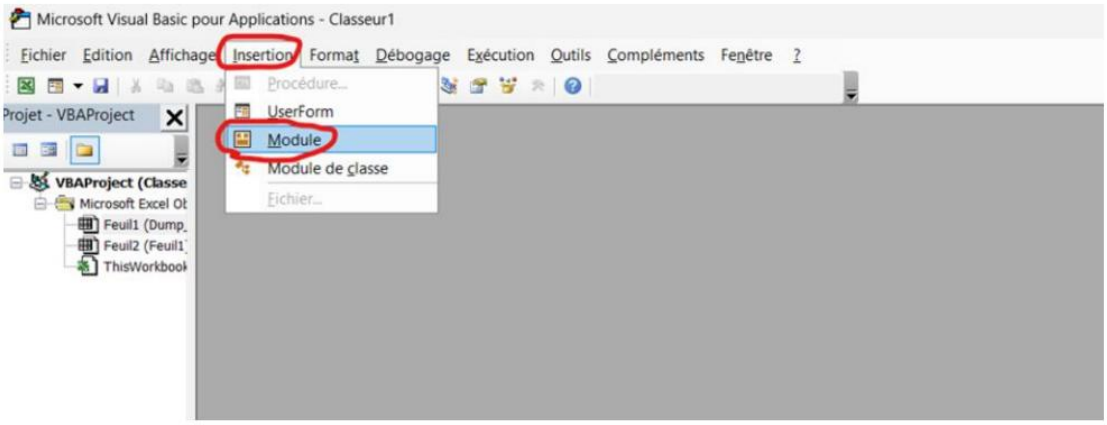
3. Install the macro:

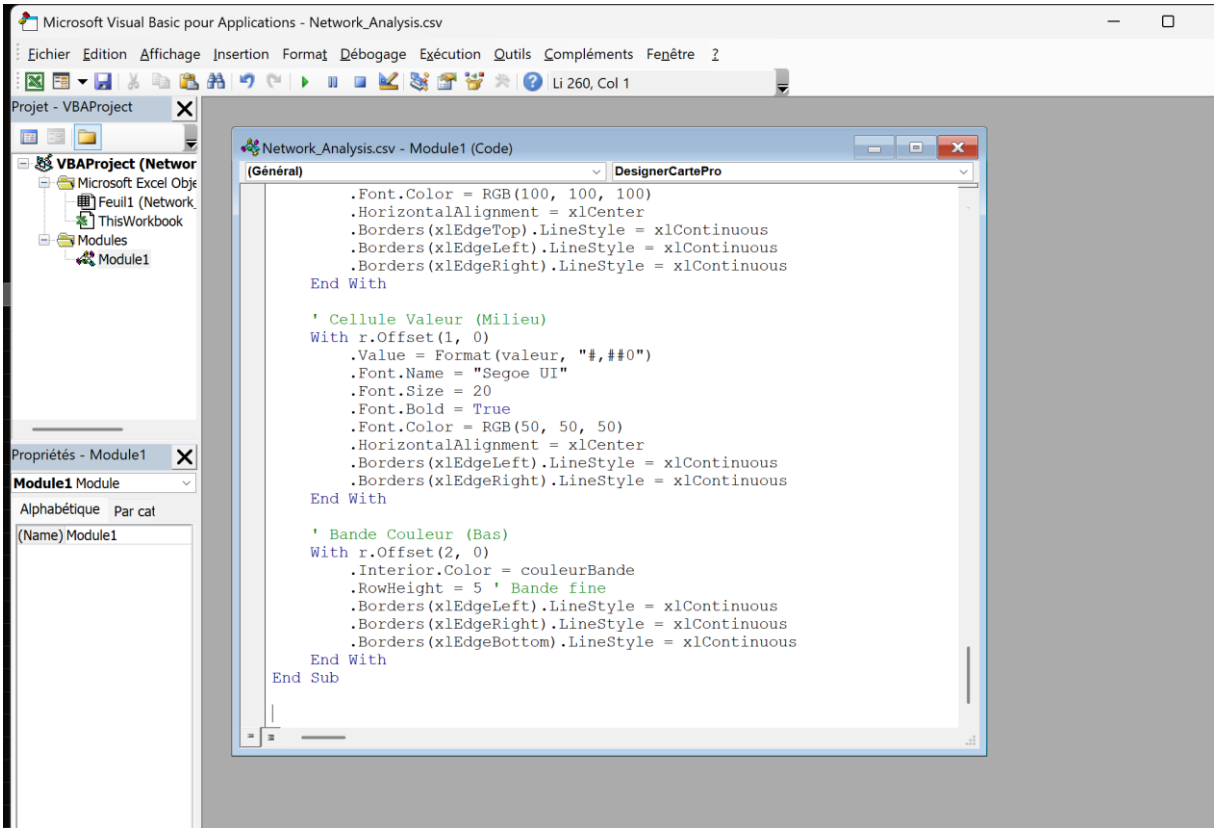
- Alt+F11 to open VBA editor

- Right click → Insert Module

- - Copy everything from the vba.txt file (available in the GitHub repository) and paste it

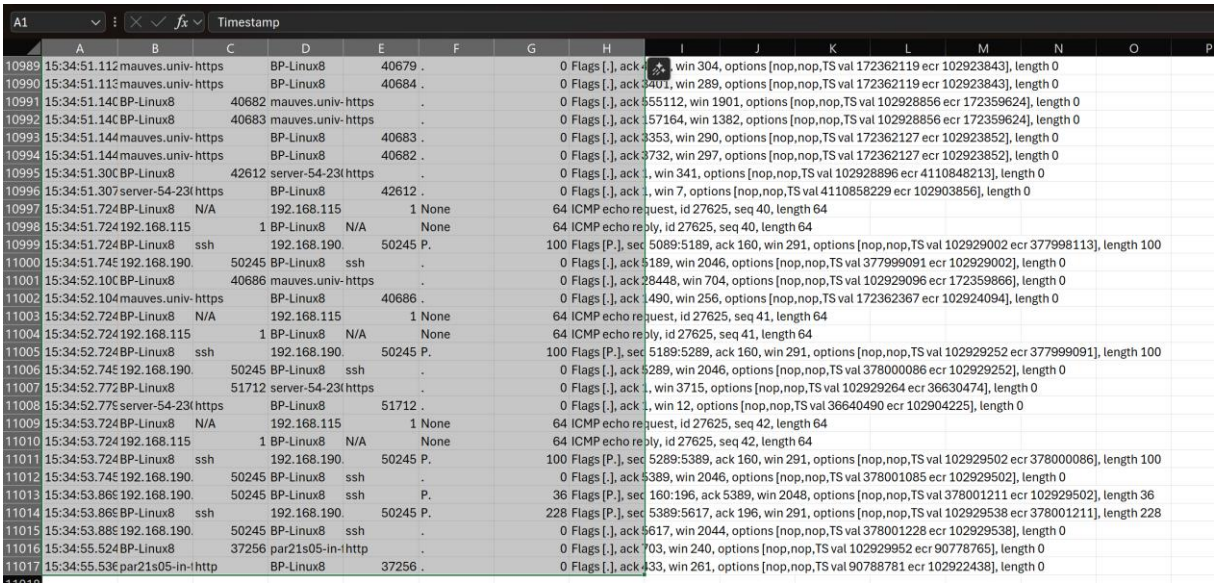
- Alt+F11 to close

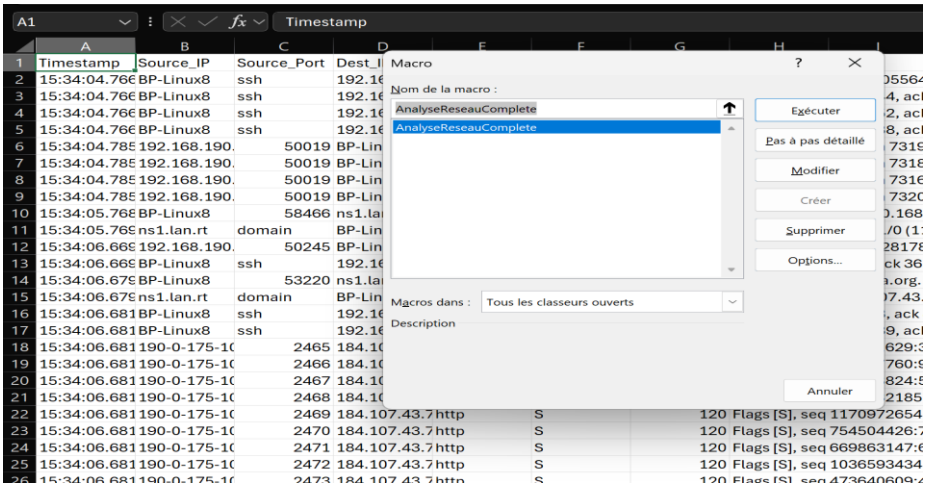




4. Run it with the macro:

- Alt+F8





- Select `AnalyserTraficReseau

- Click Exécuter

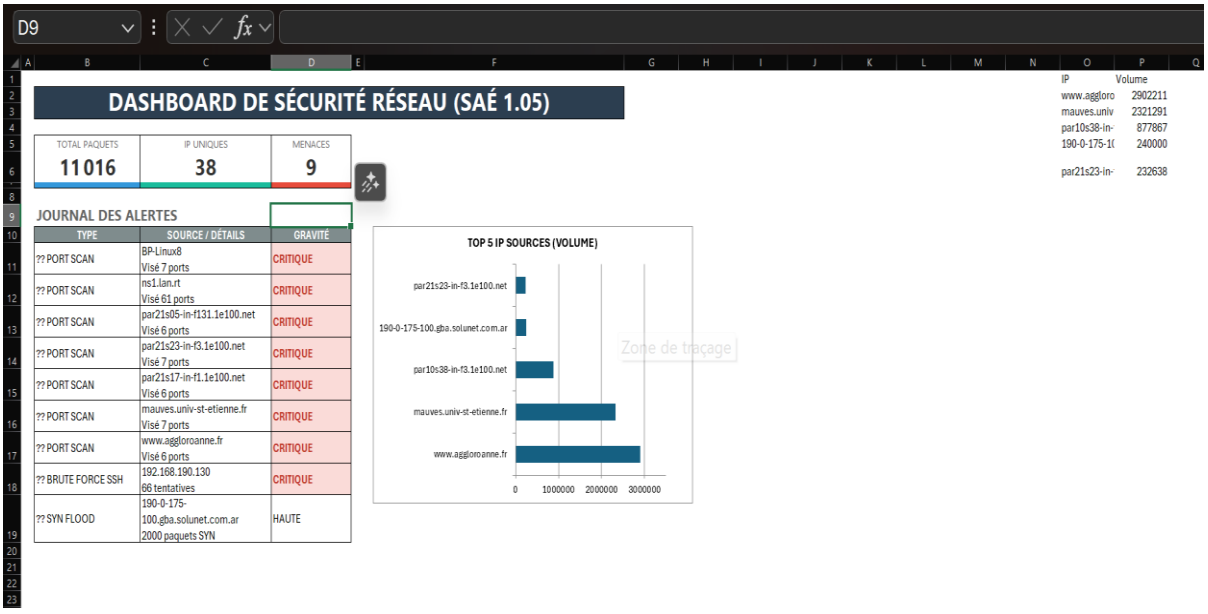
5. New sheet appears: "Analyse Réseau"

- Top 10 IPs with stats

- Protocol breakdown

- Anomalies table (if any)

- Colored cells for problems (red = issue)



Pretty useful if your boss wants an Excel report.

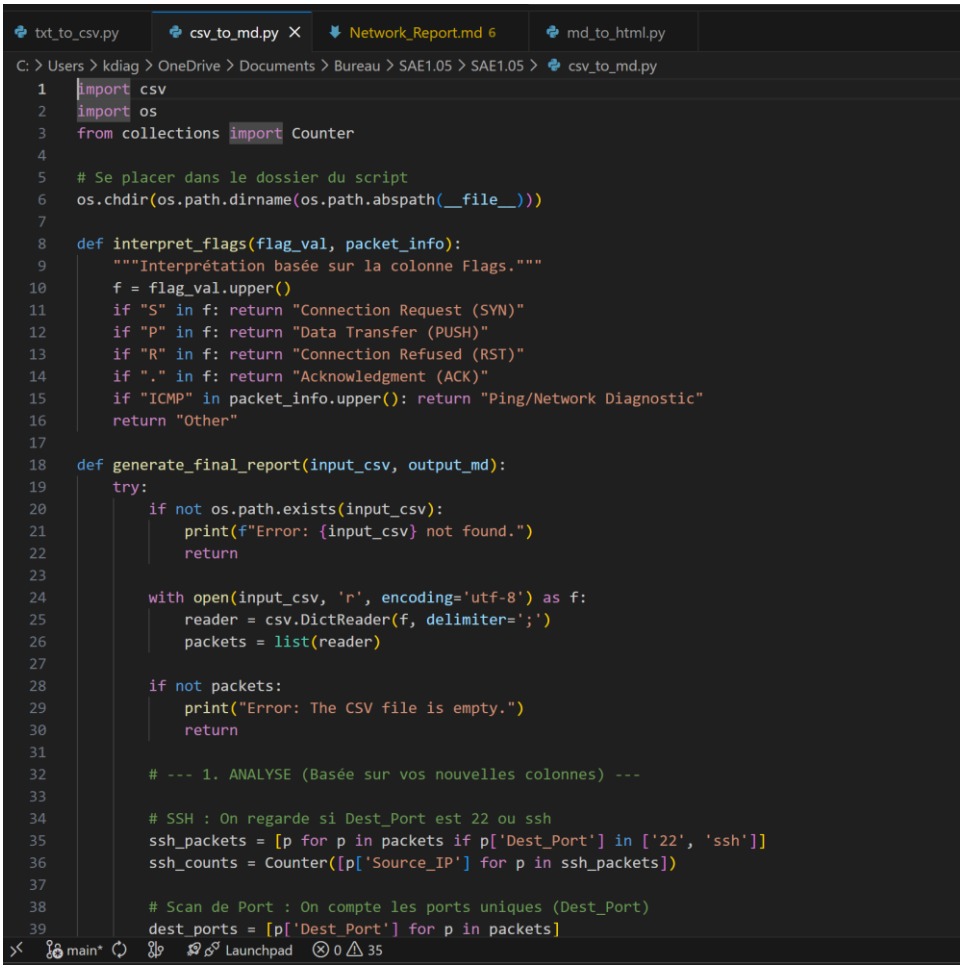
V. Phase 3: Automated Reporting (CSV to MD)

To save time and standardize security reporting, we use a second script that performs the analysis automatically.

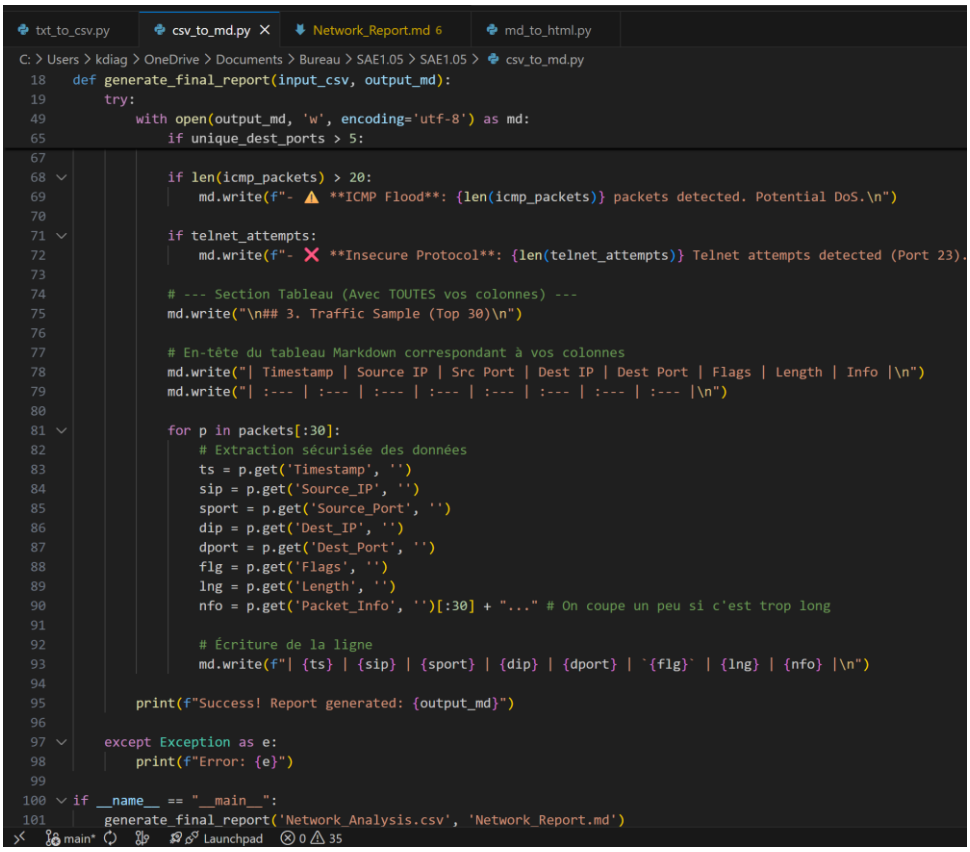
- **Execution:** Run python csv_to_md.py.
- **Process:** The script scans the CSV file, counts the connection attempts, and interprets TCP flags (SYN, PUSH-ACK, RST).
- **Output:** It generates a professional Markdown report named **Network_Report.md**.

Key Features of the Automated Report:

- **Threat Intelligence:** It automatically flags "Main Assaults" (high packet volume) and "Stealth Probes" (low packet volume).
- **Anomaly Detection:** It identifies Port Scanning (multiple ports probed) and ICMP Flooding (abnormal Ping volume).
- **Flag Interpretation:** It translates technical TCP flags into plain English for non-expert readers.



```python
import csv
import os
from collections import Counter

# Se placer dans le dossier du script
os.chdir(os.path.dirname(os.path.abspath(__file__)))

def interpret_flags(flag_val, packet_info):
    """Interprétation basée sur la colonne Flags."""
    f = flag_val.upper()
    if "S" in f: return "Connection Request (SYN)"
    if "P" in f: return "Data Transfer (PUSH)"
    if "R" in f: return "Connection Refused (RST)"
    if "." in f: return "Acknowledgment (ACK)"
    if "ICMP" in packet_info.upper(): return "Ping/Network Diagnostic"
    return "Other"

def generate_final_report(input_csv, output_md):
    try:
        if not os.path.exists(input_csv):
            print(f"Error: {input_csv} not found.")
            return

        with open(input_csv, 'r', encoding='utf-8') as f:
            reader = csv.DictReader(f, delimiter=';')
            packets = list(reader)

        if not packets:
            print("Error: The CSV file is empty.")
            return

        # --- 1. ANALYSE (Basée sur vos nouvelles colonnes) ---

        # SSH : On regarde si Dest_Port est 22 ou ssh
        ssh_packets = [p for p in packets if p['Dest_Port'] in ['22', 'ssh']]
        ssh_counts = Counter([p['Source_IP'] for p in ssh_packets])

        # Scan de Port : On compte les ports uniques (Dest_Port)
        dest_ports = [p['Dest_Port'] for p in packets]
```

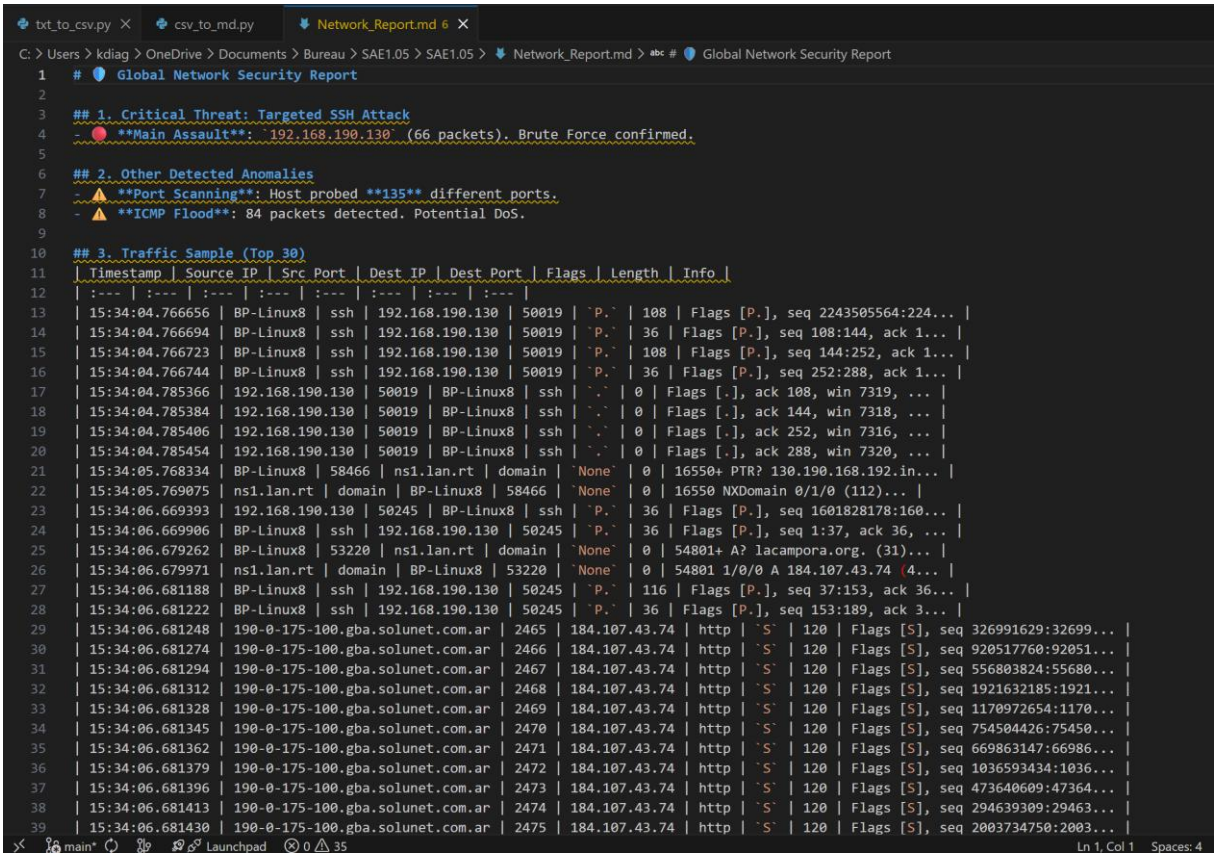


```python
def generate_final_report(input_csv, output_md):
    try:
        with open(output_md, 'w', encoding='utf-8') as md:
            if unique_dest_ports > 5:

            if len(icmp_packets) > 20:
                md.write(f"- ⚠️ **ICMP Flood**: {len(icmp_packets)} packets detected. Potential DoS.\n")

            if telnet_attempts:
                md.write(f"- ❌ **Insecure Protocol**: {len(telnet_attempts)} Telnet attempts detected (Port 23).

            # --- Section Tableau (Avec TOUTES vos colonnes) ---
            md.write("\n## 3. Traffic Sample (Top 30)\n")

            # En-tête du tableau Markdown correspondant à vos colonnes
            md.write("| Timestamp | Source IP | Src Port | Dest IP | Dest Port | Flags | Length | Info |\n")
            md.write("| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |\n")

            for p in packets[:30]:
                # Extraction sécurisée des données
                ts = p.get('Timestamp', '')
                sip = p.get('Source_IP', '')
                sport = p.get('Source_Port', '')
                dip = p.get('Dest_IP', '')
                dport = p.get('Dest_Port', '')
                flg = p.get('Flags', '')
                lng = p.get('Length', '')
                nfo = p.get('Packet_Info', '')[:30] + "..." # On coupe un peu si c'est trop long

                # Écriture de la ligne
                md.write(f"| {ts} | {sip} | {sport} | {dip} | {dport} | `{flg}` | {lng} | {nfo} |\n")

        print(f"Success! Report generated: {output_md}")

    except Exception as e:
        print(f"Error: {e}")

if __name__ == "__main__":
    generate_final_report('Network_Analysis.csv', 'Network_Report.md')
```

Network_Report.md

This file provides a detailed textual summary of the capture, including:

- Total number of packets analyzed (e.g., 11,016)
- Overall data volume statistics
- Top 30 most active IP addresses
- Protocol breakdown (TCP, UDP, ICMP)
- Security alerts (for example, 3 detected threats)



md_to_html.py

Converts Network_Report.md to a standalone Network_Report.html with:

- Embedded CSS (Bootstrap-inspired styles)
- Color-coded alerts (red/orange/green)
- Tables and sections ready for presentation or emailing

txt_to_csv.py | csv_to_md.py | Network_Report.md 6 | md_to_html.py

C: > Users > kdiag > OneDrive > Documents > Bureau > SAE1.05 > SAE1.05 > md_to_html.py

```python
import os
from pathlib import Path


BASE_DIR = Path(__file__).resolve().parent
os.chdir(BASE_DIR)


def simple_md_to_html(input_md: str, output_html: str) -> None:
    """
    Convertit un Markdown simple en rapport HTML stylé.
    """
    try:
        md_path = Path(input_md)

        if not md_path.exists():
            print(f"[ERREUR] Fichier introuvable : {md_path}")
            return

        with md_path.open("r", encoding="utf-8") as f:
            lines = f.readlines()

        html_content = []
        in_table = False

        for raw_line in lines:
            line = raw_line.strip()

            # Titres
            if line.startswith("# "):
                html_content.append(f"<h1>{line[2:]}</h1>")
            elif line.startswith("## "):
                html_content.append(f"<h2>{line[3:]}</h2>")

            # Listes
            elif line.startswith("- "):
                html_content.append(f"<li>{line[2:]}</li>")

            # Tableaux simples
```

txt_to_csv.py | csv_to_md.py | Network_Report.md 6 | md_to_html.py

C: > Users > kdiag > OneDrive > Documents > Bureau > SAE1.05 > SAE1.05 > md_to_html.py

```
466     <body>
493         <script>
495             (function() {{
513             }})();
514
515             // Horloge digitale en haut à droite
516             (function() {{
517                 const clockEl = document.getElementById('clock');
518
519                 function updateClock() {{
520                     const now = new Date();
521                     let h = now.getHours().toString().padStart(2, '0');
522                     let m = now.getMinutes().toString().padStart(2, '0');
523                     let s = now.getSeconds().toString().padStart(2, '0');
524                     clockEl.textContent = h + ':' + m + ':' + s;
525                 }}
526
527                 updateClock();
528                 setInterval(updateClock, 1000);
529             }})();
530         </script>
531     </body>
532     </html>"""
533
534             with open(output_html, "w", encoding="utf-8") as f:
535                 f.write(full_page)
536
537             print(f"[OK] Rapport généré : {output_html}")
538
539         except Exception as e:
540             print(f"[ERREUR] Une exception est survenue : {e}")
541
542
543     if __name__ == "__main__":
544         simple_md_to_html("Network_Report.md", "Network_Report.html")
545
```

Network_Report.html

This is a styled HTML version of the report, ideal for:

- Sharing by email
- Presenting during the oral defense
- Adding to a professional portfolio

Network Security Report

15:36:21

Génération automatique depuis le fichier **Network_Report.md**

Mode sombre

Global Network Security Report

1. Critical Threat: Targeted SSH Attack

- **Main Assault**: `192.168.190.130` (66 packets). Brute Force confirmed.

2. Other Detected Anomalies

- **Port Scanning**: Host probed **135** different ports.
- **ICMP Flood**: 84 packets detected. Potential DoS.

3. Traffic Sample (Top 30)

Timestamp	Source IP	Src Port	Dest IP	Dest Port	Flags	Length	Info
15:34:04.766656	BP-Linux8	ssh	192.168.190.130	50019	`P.`	108	Flags [P.], seq 2243505564:224...
15:34:04.766694	BP-Linux8	ssh	192.168.190.130	50019	`P.`	36	Flags [P.], seq 108:144, ack 1...
15:34:04.766723	BP-Linux8	ssh	192.168.190.130	50019	`P.`	108	Flags [P.], seq 144:252, ack 1...
15:34:04.766744	BP-Linux8	ssh	192.168.190.130	50019	`P.`	36	Flags [P.], seq 252:288, ack 1...
15:34:04.785366	192.168.190.130	50019	BP-Linux8	ssh	`.`	0	Flags [.], ack 108, win 7319, ...
15:34:04.785384	192.168.190.130	50019	BP-Linux8	ssh	`.`	0	Flags [.], ack 144, win 7318, ...
15:34:04.785406	192.168.190.130	50019	BP-Linux8	ssh	`.`	0	Flags [.], ack 252, win 7316, ...
15:34:04.785454	192.168.190.130	50019	BP-Linux8	ssh	`.`	0	Flags [.], ack 288, win 7320, ...
15:34:05.768334	BP-Linux8	58466	ns1.lan.rt	domain	`None`	0	16550+ PTR? 130.190.168.192.in...
15:34:05.769075	ns1.lan.rt	domain	BP-Linux8	58466	`None`	0	16550 NXDomain 0/1/0 (112)...
15:34:06.669393	192.168.190.130	50245	BP-Linux8	ssh	`P.`	36	Flags [P.], seq 1601828178:160...
15:34:06.669906	BP-Linux8	ssh	192.168.190.130	50245	`P.`	36	Flags [P.], seq 1:37, ack 36, ...
15:34:06.679262	BP-Linux8	53220	ns1.lan.rt	domain	`None`	0	54801+ A? lacampora.org. (31)...
15:34:06.679971	ns1.lan.rt	domain	BP-Linux8	53220	`None`	0	54801 1/0/0 A 184.107.43.74 (4...
15:34:06.681188	BP-Linux8	ssh	192.168.190.130	50245	`P.`	116	Flags [P.], seq 37:153, ack 36...
15:34:06.681222	BP-Linux8	ssh	192.168.190.130	50245	`P.`	36	Flags [P.], seq 153:189, ack 3...
15:34:06.681248	190-0-175-100.gba.solunet.com.ar	2465	184.107.43.74	http	`S`	120	Flags [S], seq 326991629:32699...
15:34:06.681374	190-0-175-100.gba.solunet.com.ar	2466	184.107.43.74	http	`S`	120	Flags [S], seq 920517760:92051...
15:34:06.681294	190-0-175-100.gba.solunet.com.ar	2467	184.107.43.74	http	`S`	120	Flags [S], seq 556803824:55680...
15:34:06.681312	190-0-175-100.gba.solunet.com.ar	2468	184.107.43.74	http	`S`	120	Flags [S], seq 1921632185:1921...
15:34:06.681328	190-0-175-100.gba.solunet.com.ar	2469	184.107.43.74	http	`S`	120	Flags [S], seq 1170972654:1170...
15:34:06.681345	190-0-175-100.gba.solunet.com.ar	2470	184.107.43.74	http	`S`	120	Flags [S], seq 754504426:75450...
15:34:06.681362	190-0-175-100.gba.solunet.com.ar	2471	184.107.43.74	http	`S`	120	Flags [S], seq 669863147:66986...
15:34:06.681379	190-0-175-100.gba.solunet.com.ar	2472	184.107.43.74	http	`S`	120	Flags [S], seq 1036593434:1036...
15:34:06.681396	190-0-175-100.gba.solunet.com.ar	2473	184.107.43.74	http	`S`	120	Flags [S], seq 473640609:47364...
15:34:06.681413	190-0-175-100.gba.solunet.com.ar	2474	184.107.43.74	http	`S`	120	Flags [S], seq 294639309:29463...
15:34:06.681430	190-0-175-100.gba.solunet.com.ar	2475	184.107.43.74	http	`S`	120	Flags [S], seq 2003734750:2003...
15:34:06.681446	190-0-175-100.gba.solunet.com.ar	2476	184.107.43.74	http	`S`	120	Flags [S], seq 943277646:94327...
15:34:06.681463	190-0-175-100.gba.solunet.com.ar	2477	184.107.43.74	http	`S`	120	Flags [S], seq 612921749:61292...
15:34:06.681480	190-0-175-100.gba.solunet.com.ar	2478	184.107.43.74	http	`S`	120	Flags [S], seq 1079269685:1079...

Généré automatiquement en HTML – Markdown to Report

4. Phase 4: Web Visualization (MD to HTML)

The final step of the pipeline converts the text-based report into a high-end, interactive web interface. This ensures the findings are presentable for a Security Operations Center (SOC) environment or for presentation to management.

- **Execution: Run python md_to_html.py.**
- **Process: The script parses the Markdown syntax and injects it into a custom-designed HTML5 template using Bootstrap 5.**
- **Output: A file named Network_Report.html is created, featuring a modern "Cyber Dark Mode" design.**

VI. Understanding the CSV

The generated CSV file contains **11 columns**. Here are the important ones:

- **Timestamp** – Time when the packet was sent (HH:MM:SS)
- **Protocol** – TCP, UDP, ICMP
- **Source IP** – Who sent the packet
- **Source Port** – Port used by the sender
- **Destination IP** – Where the packet is going
- **Destination Port** – Target port (e.g., 80 = HTTP, 22 = SSH, 443 = HTTPS)
- **Flags** – S = new connection, . = data, F = connection closing
- **Length** – Packet size in bytes
- **Seq** – TCP sequence number
- **Ack** – TCP acknowledgment number
- **Window** – TCP flow control value

The last three fields (Seq, Ack, Window) are mainly useful for **advanced TCP debugging**. For basic anomaly detection (SYN floods, port scans, traffic floods), you only need: **IP addresses, ports, flags, and packet length.**

VII. Common Ports (Quick Reference)

If you see these ports in the CSV:

- **22** → SSH
- **80** → HTTP

- **443** → HTTPS

VIII. Troubleshooting

- **Error: ModuleNotFoundError: matplotlib**
  → Install it with: pip install matplotlib
- **CSV appears entirely in column A**
  → In Excel: Column A → Data → Text to Columns → Delimited → Select "Semicolon"
- **Python script doesn't open**
  → Python is probably not in your PATH. Reinstall Python and check "Add to PATH".
- **Script takes too long**
  → The capture file is likely very large. Start by analyzing a 1-hour sample.
- **VBA macro error**
  → Ensure the CSV is properly imported (11 columns).
  If everything is in one column, use the "Text to Columns" fix above.

IX. Best Practices and Recommendations

Good practices

- Capture traffic during normal activity to establish a baseline
- Always keep the original .txt capture file
- Use filters to avoid capturing sensitive data

Things to avoid

- Don't panic when seeing unknown IPs — verify first
- Don't manually edit the CSV (it breaks the analysis)

For deployment

- Test the tool on a small capture first
- Adjust thresholds if you get too many false positives
- Document any changes you make

X. What to Do If You Find Issues

Serious anomalies (red)

1. Identify the IP address
2. Determine what the IP corresponds to (PC, server, external host...)
3. Contact the IT/security team with:

- The IP address
- Type of anomaly (SYN flood, port scan, etc.)
- The generated charts (e.g., anomalies.png)
- The time range concerned

Medium anomalies (orange)

1. Note the IP
2. Check if it belongs to an internal scanner (security team)
3. Monitor for 24 hours
4. Escalate if it continues

No anomalies (green)

- Good news, but still review traffic patterns
- Some issues are subtle and may not trigger thresholds
- Check the Top 30 IPs manually

XI. File Structure

```
Sae1.05_Network_Report/
├── DumpFile.txt              # Your original tcpdump file
├── Network_Analysis.csv      # Structured data (Excel-ready)
├── Network_Report.md         # 📄 Read this first! Main report
├── Network_Report.html       # 🌐 Professional styled version
├── txt_to_csv.py             # Script 1
├── csv_to_md.py              # Script 2
├── md_to_html.py             # Script 3
└── README.md                 # This file
```

Keep the structure organized — you may need to revisit older captures.

```
#### Detection thresholds

| Attack type       | Description                                      | Threshold                          |
|-------------------|--------------------------------------------------|------------------------------------|
| SSH brute force   | Numerous SSH attempts (port 22) from a single IP | > 50 SSH packets within 5 minutes  |
| Port scanning     | One IP contacts many different destination ports | > 20 distinct destination ports    |
| ICMP flood (DoS)  | Excessive ICMP packets from a single IP          | > 50 ICMP packets                  |
```

```
#### Example findings (from DumpFile.txt analysis)

- 🔴 Critical: Targeted SSH attack — Source `192.168.190.130` (66 SSH packets to port 22)
  - Severity: HIGH
  - Recommendation: block this IP and enable fail2ban or similar protection

- 🟠 Port scanning — 135 destination ports probed
  - Severity: MEDIUM
  - Recommendation: investigate the source host and check for compromise

- 🟡 ICMP flood — 84 ICMP packets detected
  - Severity: MEDIUM
  - Recommendation: apply ICMP rate limiting or filtering on edge devices

The script generates `Network_Report.md`, which includes:

- An executive summary of the capture
- Lists of suspicious IPs with associated severity levels
- Top 30 most active IPs and protocol usage statistics
- Actionable recommendations for the network/security team
```

XII. Deployment Notes (India Team)

Requirements

- Python 3.8+ and matplotlib installed
- OS: Windows 10+, Linux, macOS 10.14+
- Admin rights required
- At least 4 GB RAM for large captures

Installation (Linux example)

sudo apt update

sudo apt install python3 python3-pip -y pip3 install matplotlib python3 --version pip3 list | grep matplotlib

File transfer

Clone the repository:

git clone https://github.com/kdiagne0799/Sae1.05_Network_Report.git

Testing

python Extraction txt_to_csv.py python Extraction `csv_to_md`. py

Notes

- Start with a small capture
- Input must be a tcpdump .txt file
- Results are saved in resultats_interface/
- Adjust thresholds if needed

Contact

- Technical support: khadim.diagne@etu.univ-st-etienne.fr
- GitHub: https://github.com/kdiagne0799/Sae1.05_Network_Report.git

Final Notes

This tool is not meant to replace full security monitoring. It provides a **starting point** for diagnosing network issues.

Always:

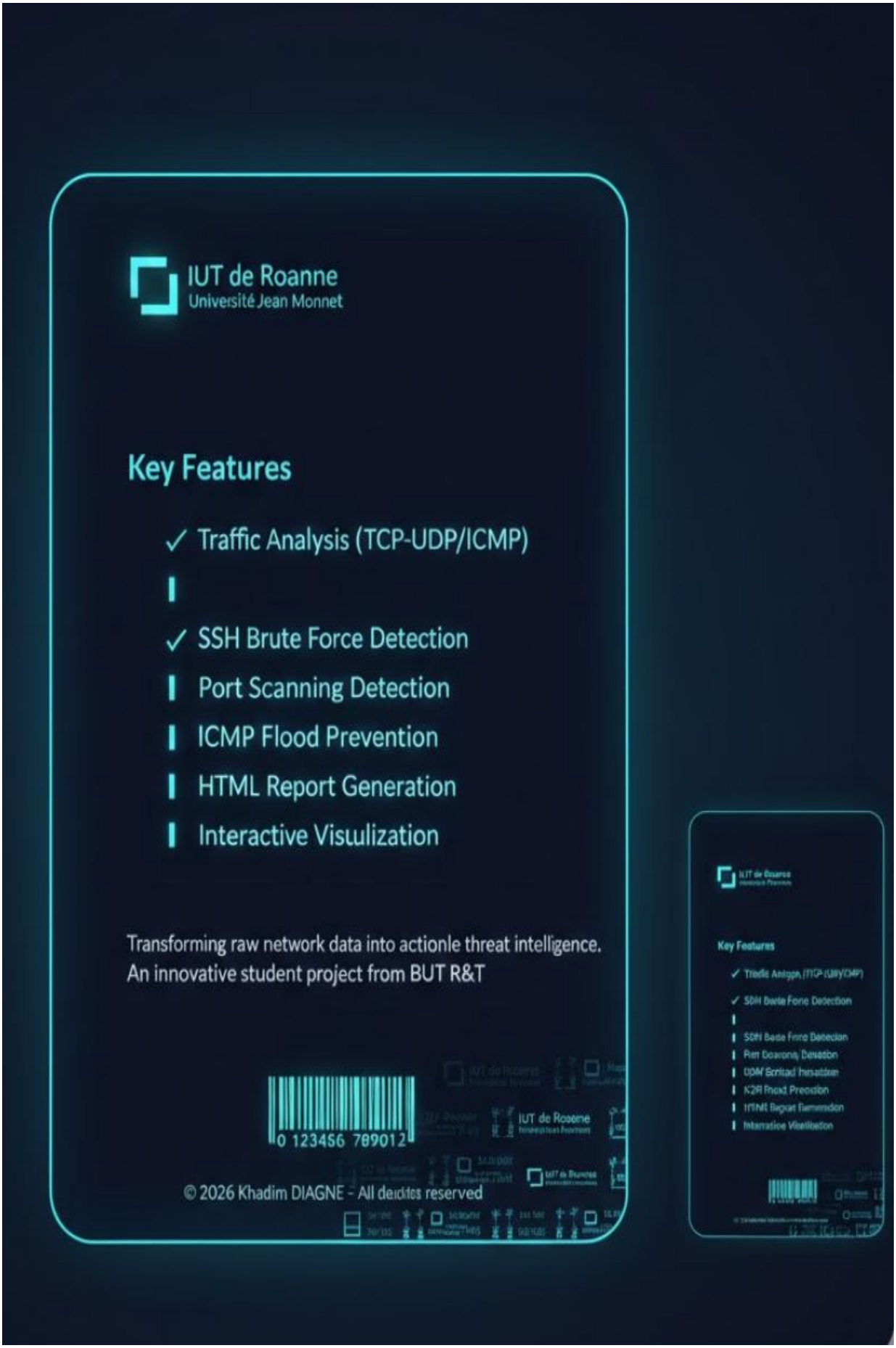
- Cross-check with IDS/firewall logs
- Verify suspicious IPs before blocking
- Document your findings
- Keep capture files for future reference

Thresholds may need tuning depending on your network environment.

Good luck — hope this helps resolve the saturation issue.
v1.0 – Jan 2026 (SAE 1.05 – BUT R&T1)

XIII. *Conclusion*

This user guide has presented the main steps to install, run and interpret the Network Analysis Tool, from raw traffic capture to an HTML report. By following these procedures, operators can more easily detect suspicious traffic patterns and document incidents. This tool is not a replacement for a full SIEM or IDS, but a lightweight assistant that can be integrated into existing workflows and improved in future versions.


IUT de Roanne
Université Jean Monnet
Key Features
✓ Traffic Analysis (TCP-UDP/ICMP)
✓ SSH Brute Force Detection
Port Scanning Detection
ICMP Flood Prevention
HTML Report Generation
Interactive Visulization
Transforming raw network data into actionle threat intelligence.
An innovative student project from BUT R&T
0 123456 789012
© 2026 Khadim DIAGNE - All deidtes reserved