Detecting Malicious Patterns through HTTP Request and Response Analysis

BY

Ajayi, Emmanuel Oluwagbenga

September, 2025

Table of Contents

Executive Summary

This project analyzes HTTP traffic from a public PCAP to detect malicious activity using Wireshark. The analysis focused on HTTP requests and responses and identified automated reconnaissance and probing attempts targeting sensitive resources (e.g., `.env`, AWS credential endpoints, Symfony config). Although most probes returned 404s, repeated attempts from multiple external IPs and a POST attempt to deliver a Mozi binary indicate hostile scanning and exploitation attempts. Recommendations include blocking identified IPs, restricting access to sensitive files, and deploying WAF rules.

Scope & Objectives

The primary objective of this project is to:

Analyze HTTP traffic within the provided PCAP file.

Identify suspicious or malicious request patterns (e.g., repeated probing, unusual POST/GET requests).

Investigate HTTP response codes for anomalies.

Document findings and potential Indicators of Compromise (IOCs).

Provide actionable recommendations for mitigating similar HTTP-based attacks.

Tools & Methodology

Wireshark was used to analyze captured network traffic from a PCAP dataset obtained via Malware Traffic Analysis ("Three Days of Scans, Probes, and Web Traffic"). The investigation followed a structured process to identify threats and anomalies.

Tools Used

Wireshark: for packet inspection and filtering

PCAP Dataset: "Three Days of Scans, Probes, and Web Traffic" (Malware Traffic Analysis)

Methodology

PCAP Import: Loaded the PCAP file into Wireshark.

Filtering HTTP Traffic: Applied filters such as:

http.request → to analyze incoming requests.

http.response → to review server responses.

http.response.code == 200/301/302/403/404 → to check for success, redirects, forbidden attempts, and failed requests.

Suspicious Request Detection: Looked for abnormal patterns such as:

Access attempts to sensitive files (/.env).

Injection-style payloads (wget commands in POST requests).

High volumes of repeated requests from the same IP addresses.

Correlation & Grouping: Grouped IPs with repeated suspicious behavior.

Documentation: Captured screenshots, summarized findings, and extracted IOCs.

Initial Traffic Overview

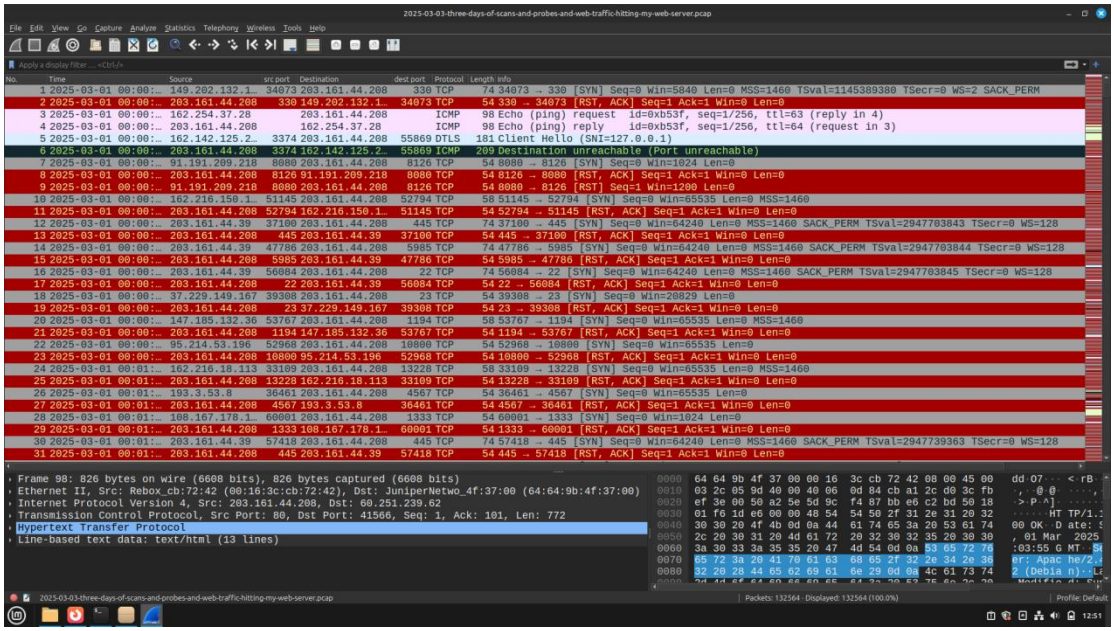

Fig 1: Initial Traffic Capture (unfiltered)

I examined the raw packet capture before applying any filters. The capture file used is:

File Name: *2025-03-03-three-days-of-scans-and-probes-and-web-traffic-hitting-my-web-server.pcap*

File Size: *3.9 MB*

Capture Duration: *3 days*

Results Snapshot

Metric	Value
Total packets (PCAP)	132,564
HTTP packets (filter: `http`)	3,823
HTTP requests (`http.request`)	1,879
HTTP responses (`http.response`)	1,891
HTTP 200 (`http.response.code == 200`)	337
HTTP 404 (`http.response.code == 404`)	1,435
Notable malicious probes	`.env`,`/.aws/credentials`, `parameters.yml`, GPON/Mozi POST

HTTP Traffic Analysis

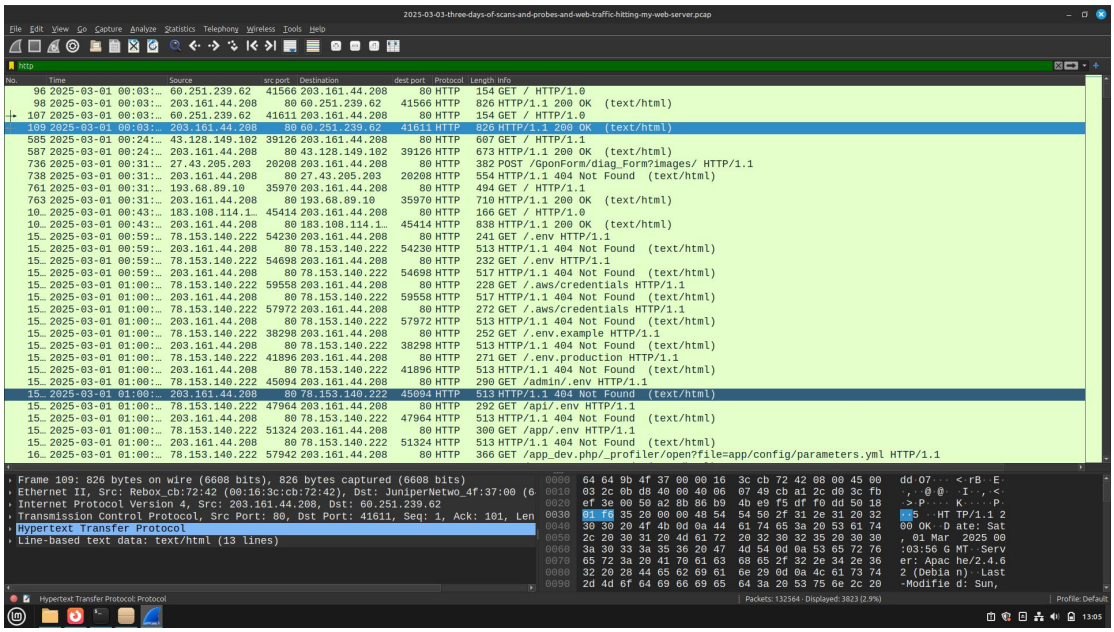

Fig 2 : Filtered HTTP Packets (3,823 packets)

To focus on web-based communication, I applied the filter: **http**

This revealed a total of 3,823 HTTP packets within the capture. These packets represent different client-server interactions such as HTTP requests, responses, and potential scanning activity.

The analysis provides insights into:

Request Methods: (e.g, GET, POST)

Response Codes: (eg, 200 OK, 404 Not Found, 302 Redirect)

Host and URI Details: Identifying requested domains and resources.

HTTP Requests

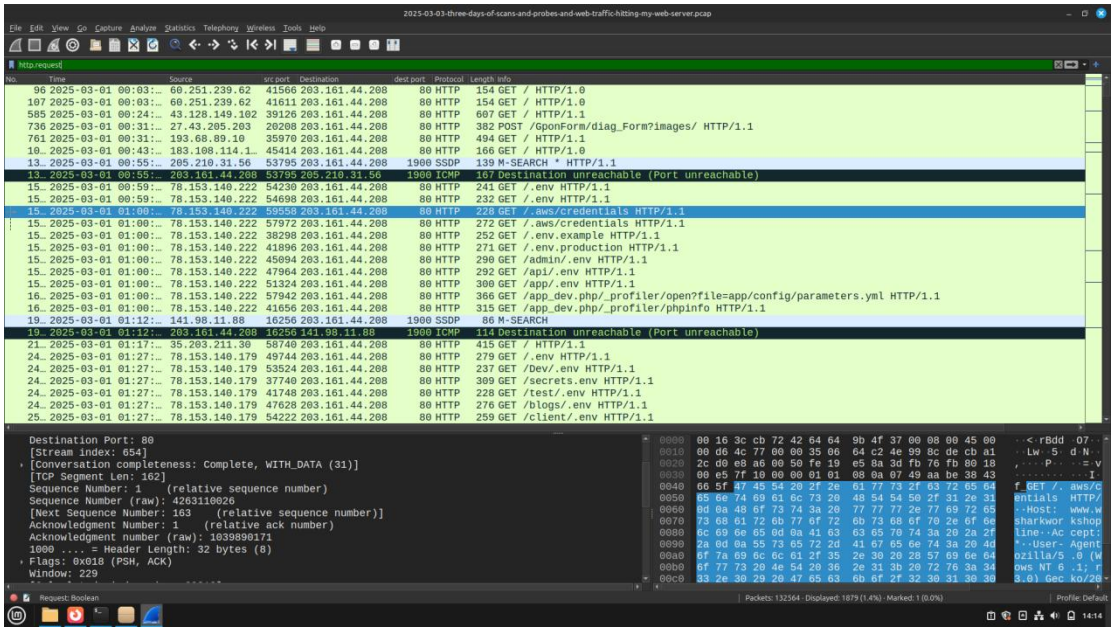

Fig 3: HTTP Requests Overview (filter: http.request)

The screenshot above provides an overview of all captured HTTP requests. A total of 1,879 HTTP request packets were recorded, forming the basis for identifying normal browsing patterns as well as potential anomalies within the traffic.

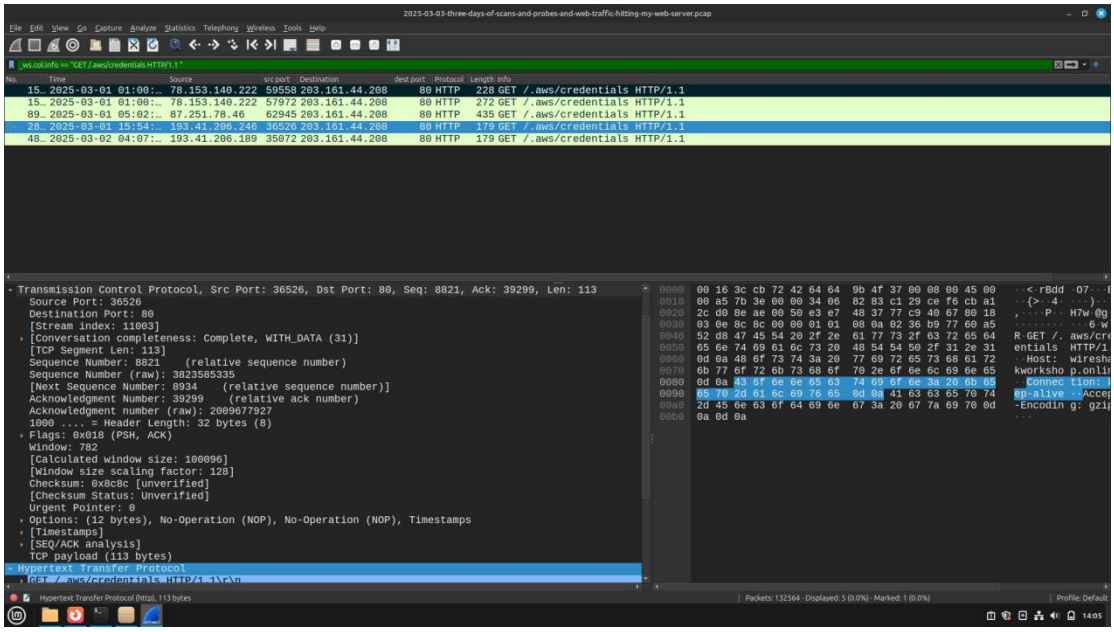

Fig 4: Suspicious GET for AWS Credentials

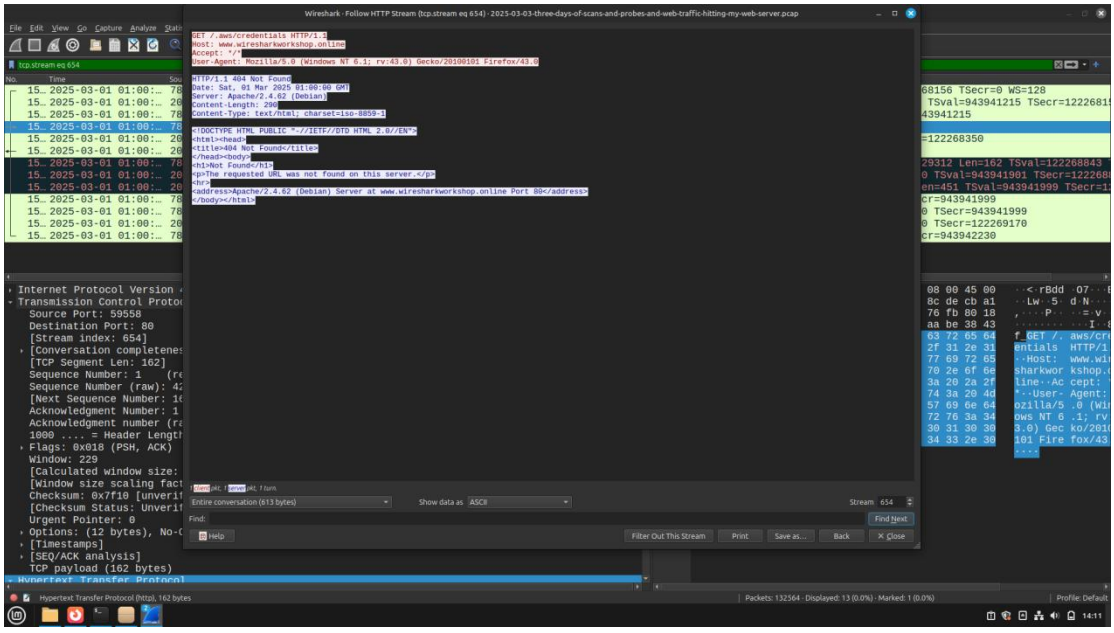

Fig 5: Conversation of the AWS GET Request

During HTTP request analysis, unusual GET requests targeting AWS credential resources were identified. Three different IP addresses (78.153.140.222, 87.251.78.46, 193.41.206.189) attempted to retrieve /.aws/credentials via HTTP. The server responded with 404 errors, indicating

no exposure. However, this pattern suggests reconnaissance activity targeting potential cloud credential files.

Suspicious HTTP Requests – .env File

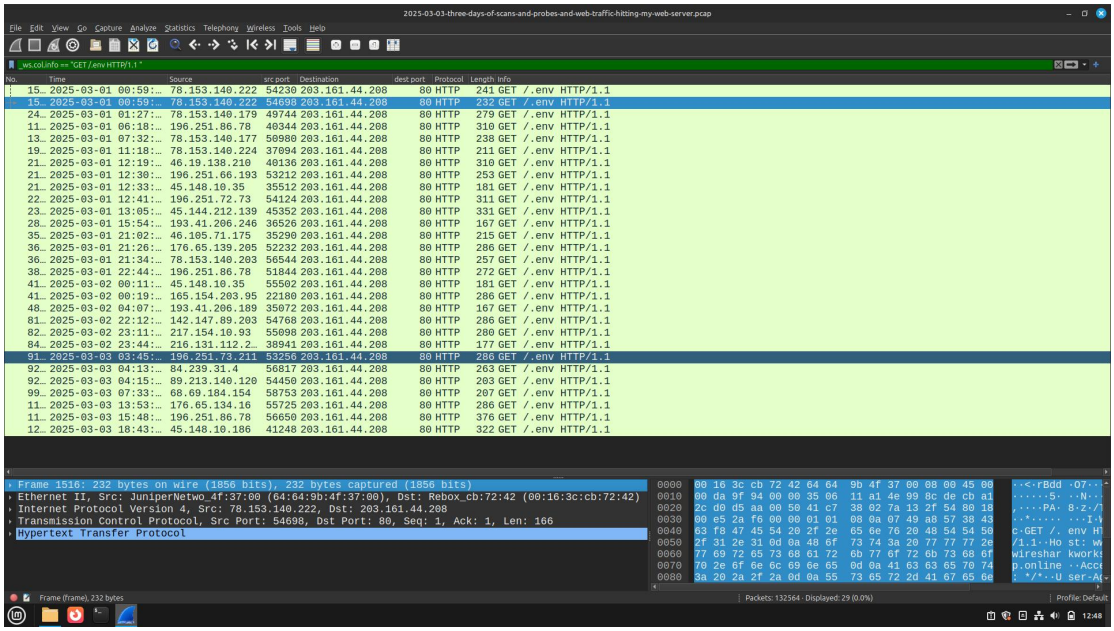

Fig 6: .env Probing Summary (29 unique source IPs)

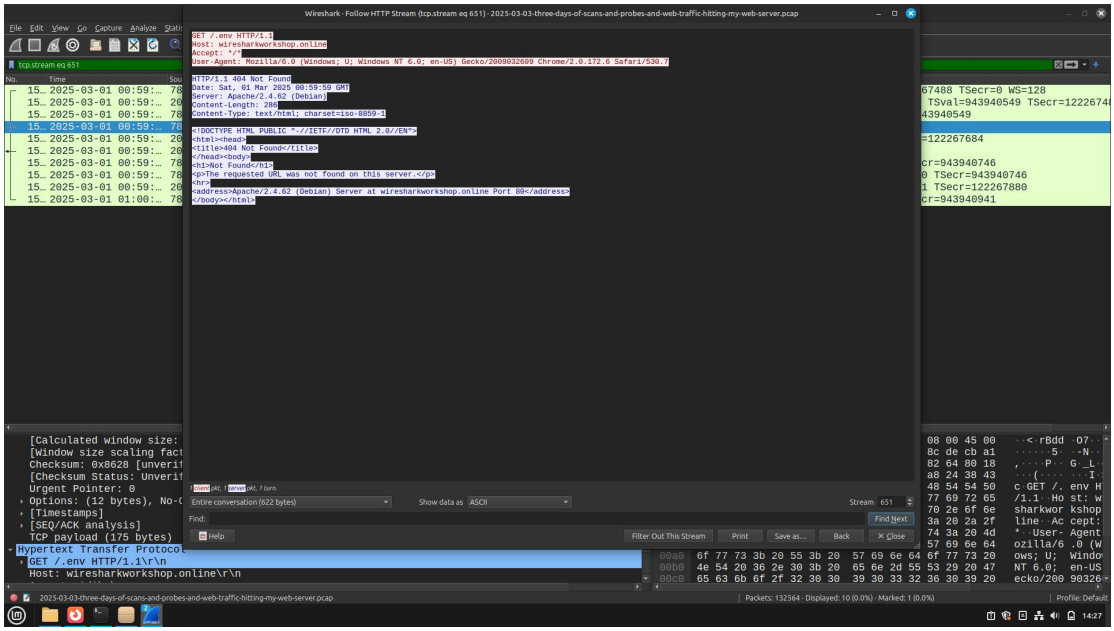

Fig 7: HTTP Stream of /.env Request and 404 Response

Observation:

29 unique IPs attempted to access the /.env file, a common target for credentials. All requests returned 404 Not Found.

Assessment:

This indicates automated scanning with malicious intent.

Sensitive File Probing – /admin/.env

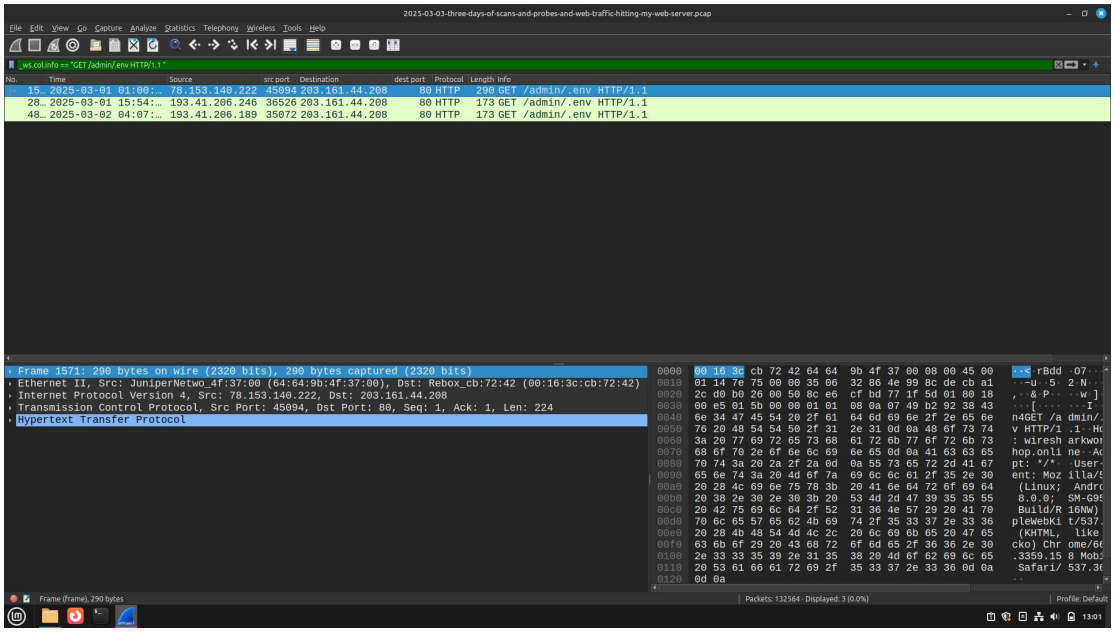

Fig 8: IPs Probing /admin/.env

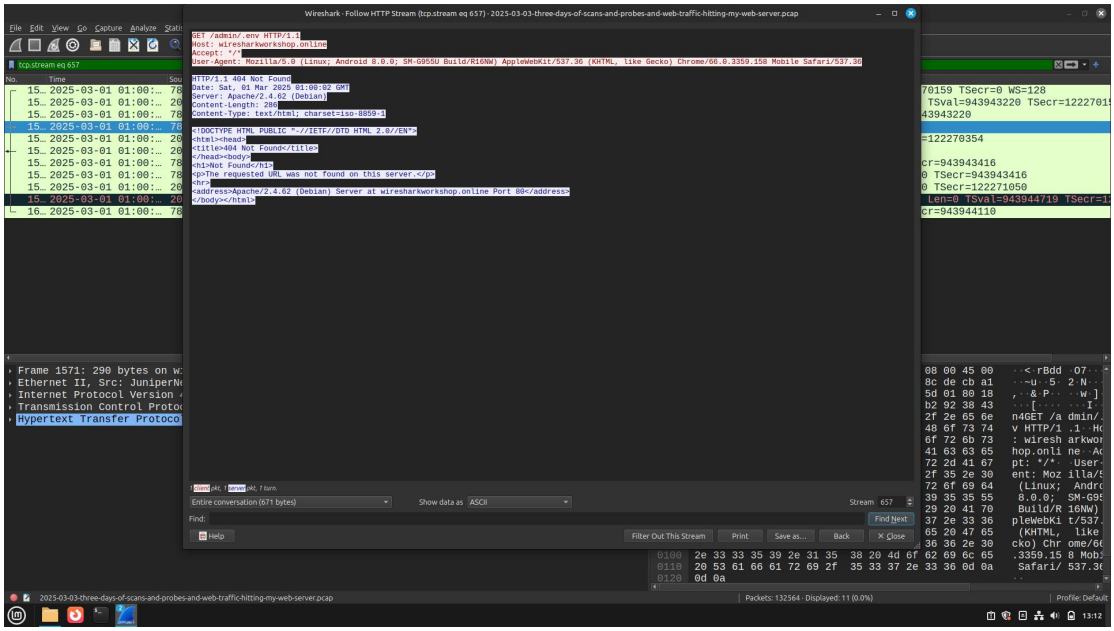

Fig 9: Conversation – GET /admin/.env Request

Observation:

Three unique IP addresses (78.153.140.222, 193.41.206.246, 193.41.206.246) attempted to access /admin/.env, a hidden environment file that typically stores credentials and API keys. All requests returned 404 Not Found.

Assessment:

This activity is malicious, indicating automated scanning or targeted probing for sensitive configuration files.

Sensitive File Probing parameters.yml

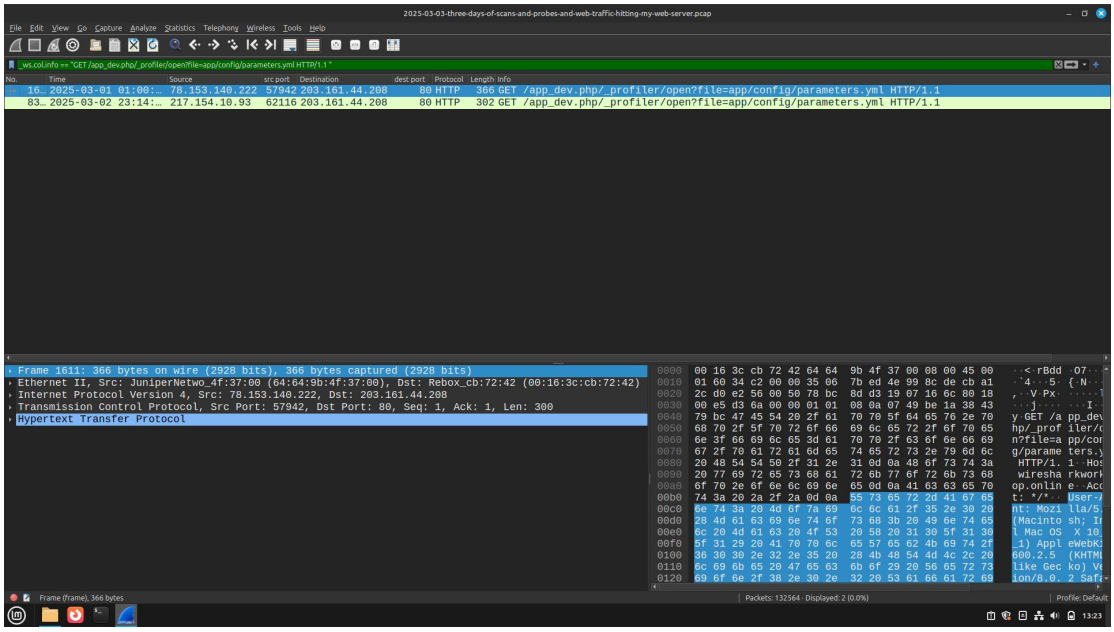

Fig 10:IPs Probing for parameters.yml

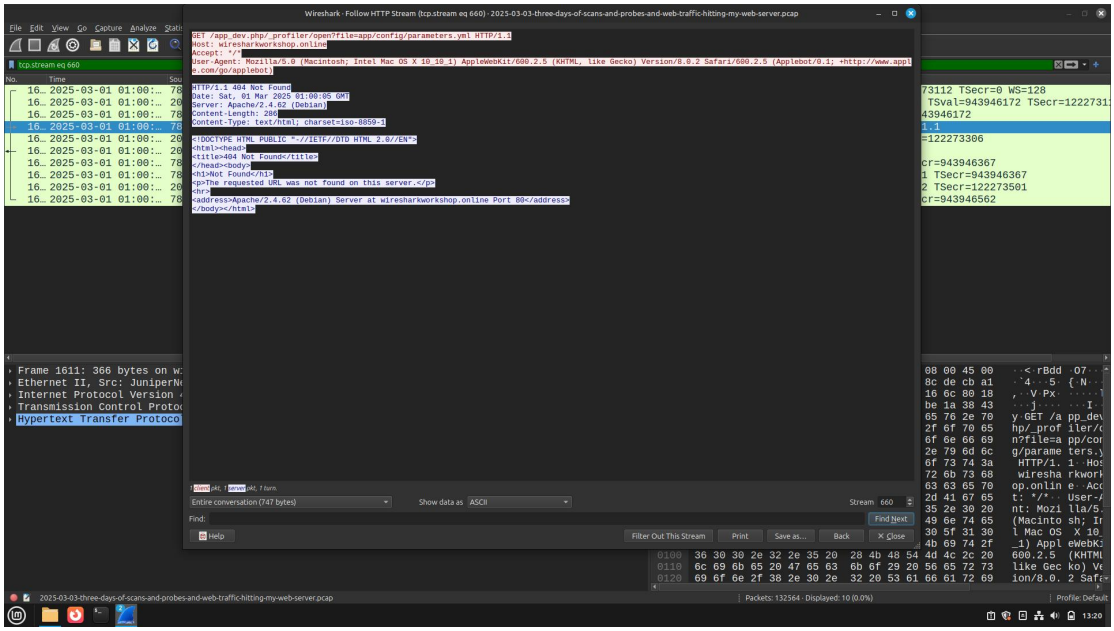

Fig 11: Conversation for Config File Probe

Observation:

Two unique IP addresses(78.153.140.222, 217.154.112.93) attempted to access /app_dev.php/_profiler/open?file=app/config/parameters.yml, which is linked to Symfony framework configuration files. Both requests received a 404 Not Found response.

Assessment:

Malicious probing for sensitive application configuration. This indicates automated reconnaissance. The User-Agent (Applebot) is likely spoofed.

Conclusion

The HTTP traffic analysis revealed multiple malicious probes, including attempts to access sensitive files such as AWS credentials, .env, and configuration files. These activities were initiated by several unique external IPs, indicating automated reconnaissance targeting the server.

Recommendations

Block or blacklist malicious IPs identified during analysis.

Monitor for repeated access attempts from the same IP ranges.

Restrict public access to sensitive files and enforce least-privilege access.

Deploy WAF and intrusion detection/prevention systems for proactive defense.

HTTP Response

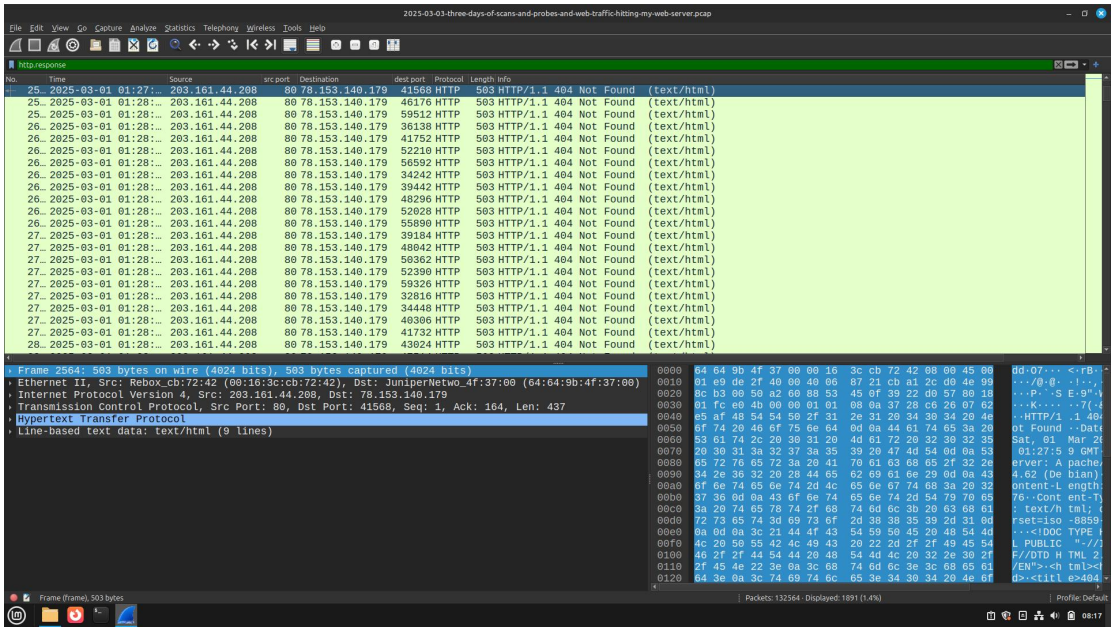

Fig 12: HTTP Responses Overview

A total of 1,891 HTTP response packets were captured. The responses consist of various status codes, primarily indicating normal web server activity. This dataset forms the basis for analyzing successful responses, errors, and potential signs of malicious probing.

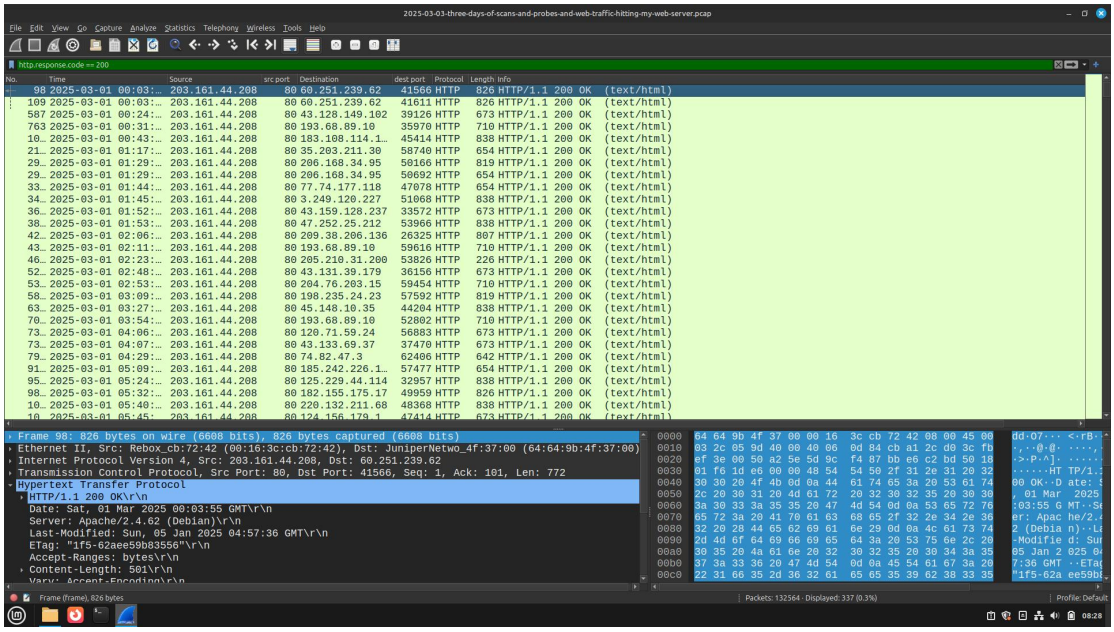

Fig 13: HTTP 200 OK Responses

Using the filter **http.response.code == 200**, a total of 337 HTTP response packets were captured. All responses consistently returned text/html content, serving the same webpage.

No malicious payloads observed. Traffic indicates normal web server responses.

HTTP Redirection Responses (301/302)

Observation

Filter applied: **http.response.code == 301 || http.response.code == 302**. No packets were captured.

No HTTP redirection activity observed during the capture period.

HTTP Forbidden Responses (403)

Observation

Filter applied: **http.response.code == 403**. No packets were captured.

No HTTP forbidden responses detected during the capture period.

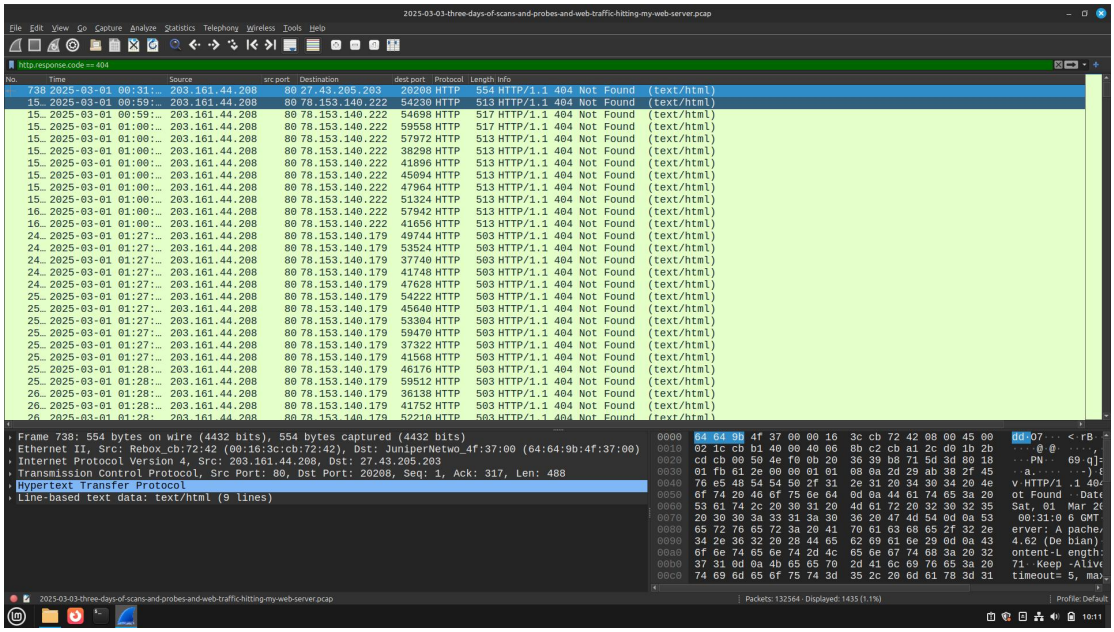

Fig 14: HTTP 404 Not Found Overview

Filter: **http.response.code == 404**

Observation: 1,435 HTTP responses from 203.161.44.208 returned 404 Not Found (text/html) to many external IPs (heavy repetition to a few IPs visible).

Assessment: Pattern is consistent with automated path enumeration / reconnaissance. The default 404 page also leaks server banner (Apache/2.4.62 (Debian)), aiding fingerprinting.

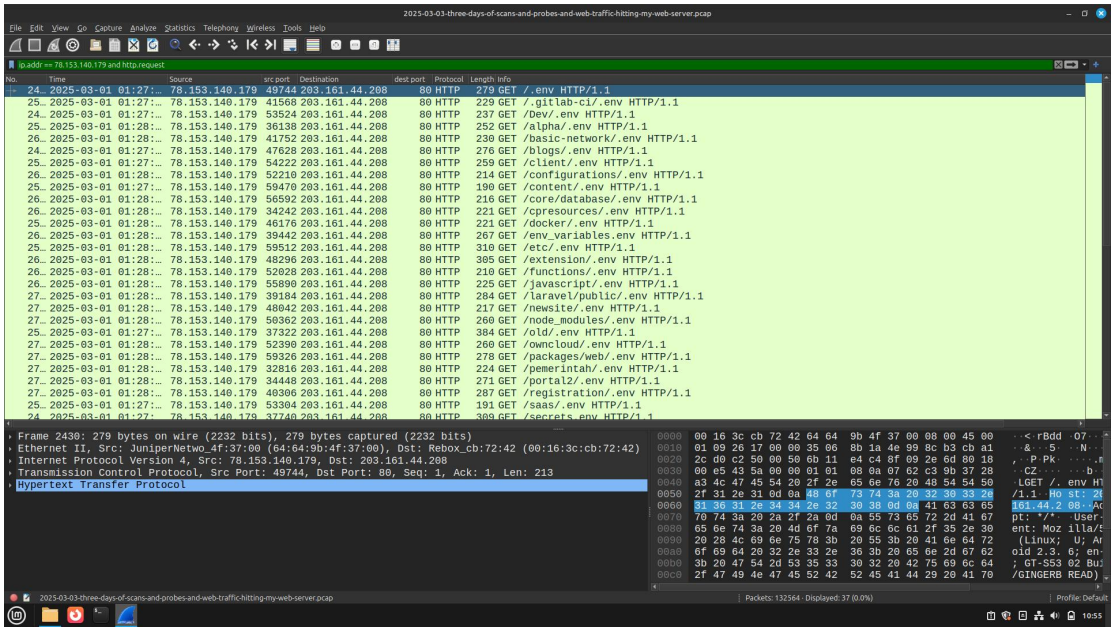

Fig 15: Repeated Probing from 78.153.140.179

The source IP 78.153.140.179 generated 37 HTTP GET requests targeting many .env file locations (e.g. /.env, /.gitlab-ci/.env, /docker/.env, /laravel/public/.env, /.secrets.env.). All requests returned 404 Not Found.

This is automated reconnaissance focused on environment/config files (credential harvesting). Although no file was retrieved, the behavior is malicious and indicates active scanning for exposed secrets.

Filter used: **ip.addr == 78.153.140.179 and http.request**.

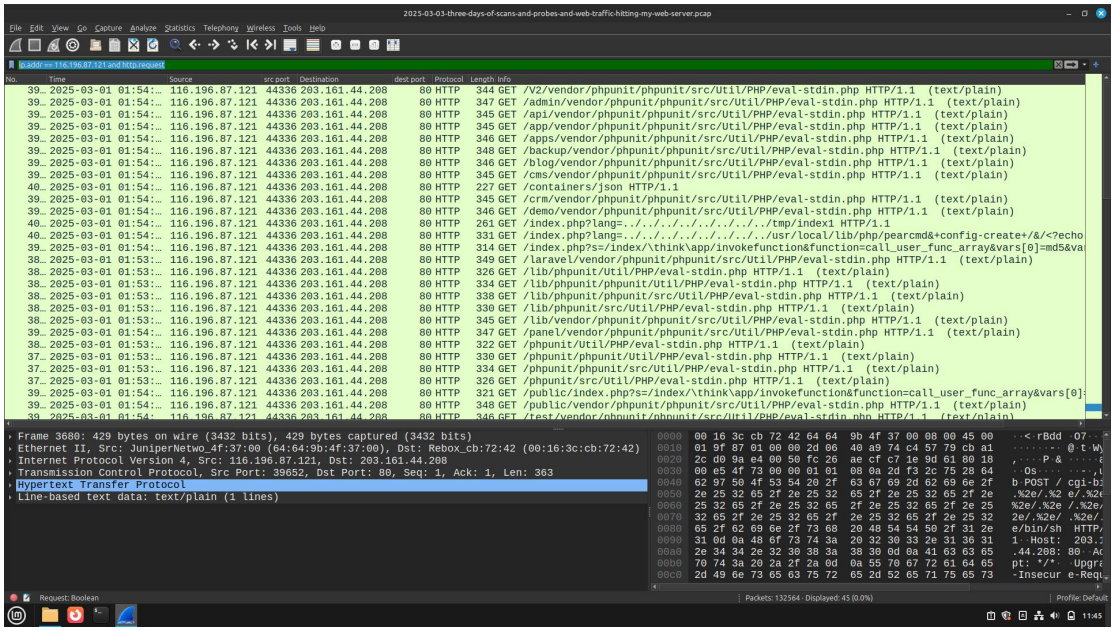

Fig 16: Probing from 116.196.87.121

Source IP 116.196.87.121 issued 45 HTTP requests targeting many PHP/framework paths (notably phpunit eval-stdin endpoints and ThinkPHP-style index.php?s=...invokefunction payloads).

Behavior is automated exploitation scanning (attempts to trigger remote code execution in common web frameworks/tools). No confirmed compromise in the capture (responses appear to be 404), but the activity is high-risk.

Filter used: **ip.addr == 116.196.87.121 and http.request**

In addition to the IPs highlighted, several other IPs were observed making repeated attempts to access .env and other sensitive files. This indicates a coordinated automated scanning campaign rather than isolated incidents.

Indicators of Compromise (IOCs)

The following Indicators of Compromise (IOCs) were extracted and enriched using VirusTotal.

IP Address	Behavior in Traffic	VT Detection Score	Verdict	Appendix Ref
78.153.140.179	Repeated .env file probes (37 requests)	13/95 flagged	Malicious	Fig A1
116.196.87.121	Multiple phpunit/ThinkPHP exploit requests (45)	0/95 flagged	Suspicious Scanning	Fig A2
87.251.78.46	Attempted access to /.aws/credentials	6/95 flagged	Malicious	Fig A3
193.41.206.189	Probing for /.aws/credentials	4/95 flagged	Malicious	Fig A4
217.154.112.93	Attempted access to Symfony parameters.yml	0/95 flagged	Suspicious Scanning	Fig A5

Suspicious URIs & Context

The following URI patterns were observed in HTTP requests and are commonly associated with credential harvesting, application exploitation, or malware delivery. Each entry includes a short rationale and mapping to MITRE ATT&CK and known public references.

Suspicious URI / Pattern	Why it's suspicious	MITRE ATT&CK Technique(s)	Known CVE / Reference
/.env, /admin/.env, /javascript/.env	Environment files often contain secrets (DB creds, API keys). Attackers probe these to harvest credentials.	**T1552** (Unsecured Credentials)	—
/.aws/credentials	Direct attempt to retrieve cloud credentials (AWS). High value for account takeover.	**T1552** (Unsecured Credentials)	—
/app_dev.php/_profiler/open?file=app/config/parameters.yml	Attempts to read framework config files (Symfony parameters.yml) which may contain secrets.	**T1552** (Unsecured Credentials); **T1190**	—

Suspicious URI / Pattern	Why it's suspicious	MITRE ATT&CK Technique(s)	Known CVE / Reference
		(Exploit Public-Facing Application)	
/vendor/phpunit/.../eval-stdin.php	Known PHPUnit debug endpoint abused to achieve remote code execution on vulnerable installs.	**T1190** (Exploit Public-Facing Application)	**CVE-2017-9841** (PHPUnit eval-stdin RCE)
index.php?s=/Index/...invokefunction&function=call_user_func_array... (ThinkPHP style)	Typical ThinkPHP RCE attempt to invoke functions remotely — remote code execution pattern.	**T1190** (Exploit Public-Facing Application)	**CVE-2019-9082** (ThinkPHP RCE class of exploits)
POST /GponForm/diag_Form?...wget http://.../Mozi.m	Command-injection attempt to download/execute a payload (Mozi botnet sample). Represents malware delivery via HTTP.	**T1105** (Ingress Tool Transfer); **T1190** (Exploit Public-Facing Application)	Mozi botnet references (public analysis)

Conclusion

The HTTP traffic analysis revealed consistent reconnaissance and probing activity against the target server. Multiple external IPs attempted to access sensitive files such as .env, AWS credentials, and Symfony configuration files. Exploit paths related to PHPUnit and ThinkPHP were also observed, alongside a POST request attempting to deliver a Mozi botnet binary.

Although all malicious requests resulted in HTTP 404 responses (unsuccessful), the repeated patterns clearly indicate automated scanning and exploitation attempts. This behavior demonstrates that the server was actively targeted, even if no compromise occurred within the captured dataset.

Recommendations

Block or Monitor Malicious IPs: Add identified source IPs to firewall blocklists or intrusion detection watchlists.

Harden Web Applications: Restrict access to sensitive files (.env, parameters.yml, .aws/credentials) through proper file permissions and server configuration.

Deploy WAF Rules: Implement Web Application Firewall signatures to detect and block exploit attempts such as PHPUnit RCE and ThinkPHP invocation patterns.

Reduce Information Exposure: Disable or limit server error messages and directory listings to minimize reconnaissance opportunities.

Continuous Monitoring: Feed these IOCs into a SIEM to detect repeat activity and correlate with other attack vectors.

Appendix

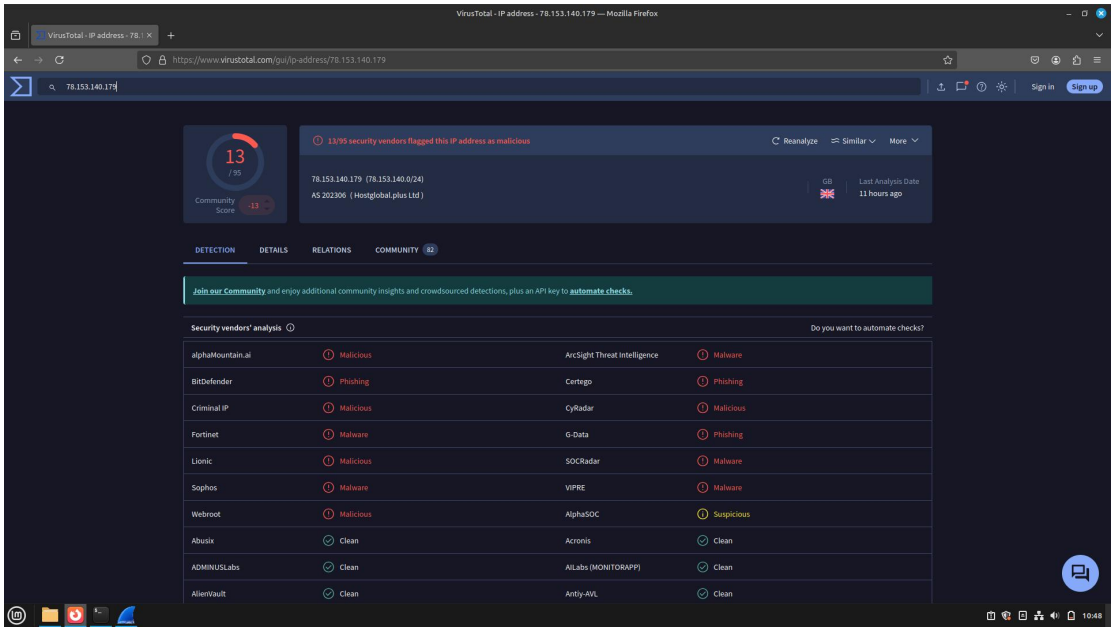

Fig A1: VirusTotal Analysis of IP 78.153.140.179

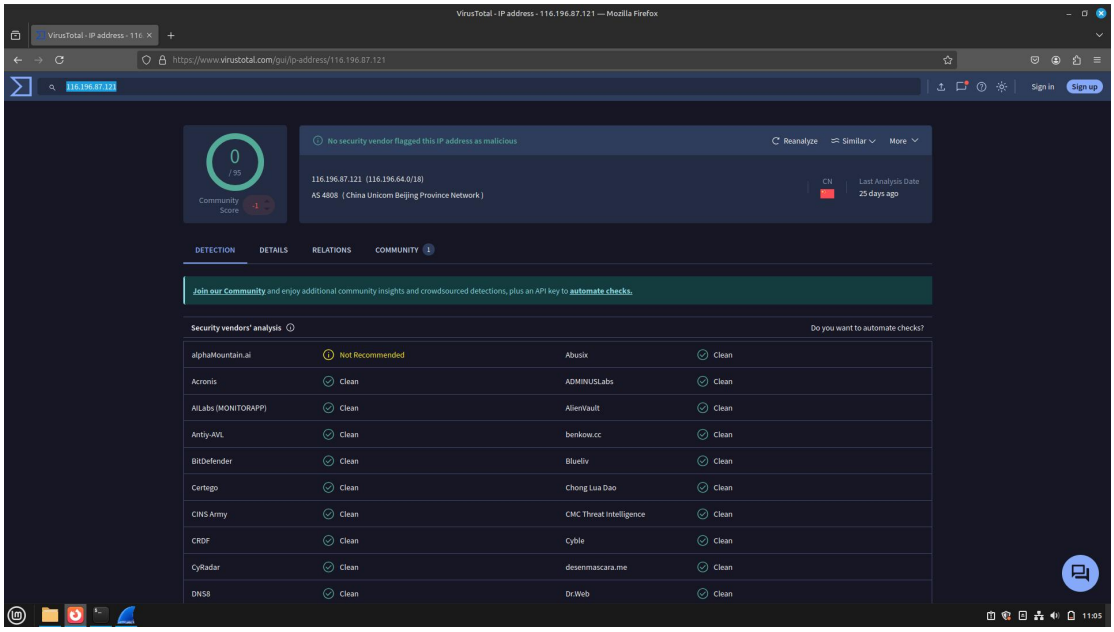

Fig A2: VirusTotal Analysis of IP 116.196.87.121 (No detections, but suspicious behavior observed in traffic).

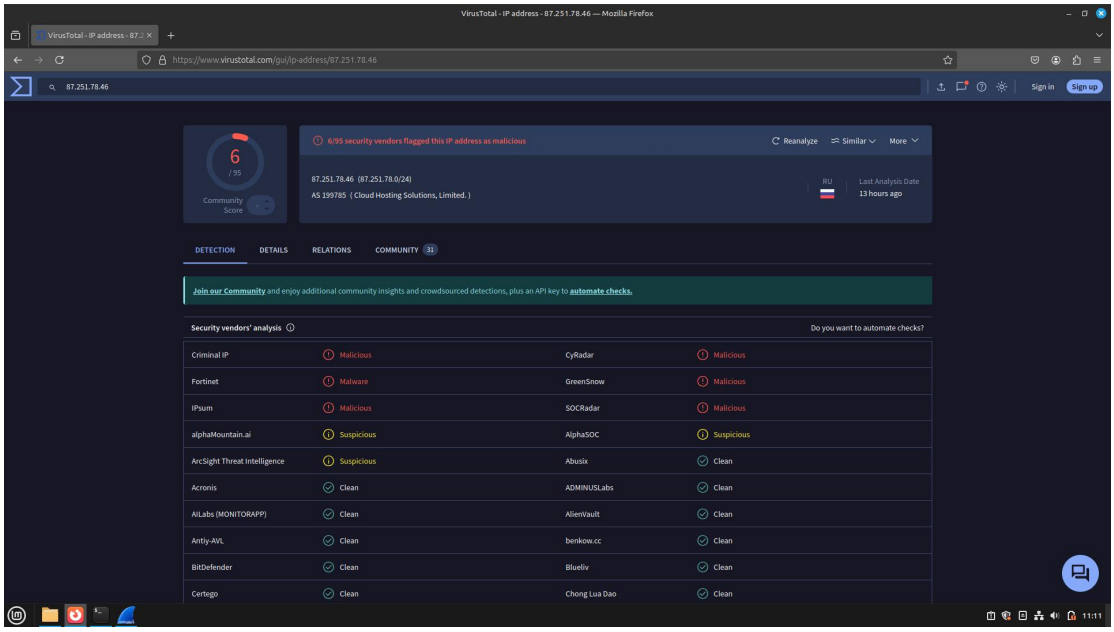

Fig A3: VirusTotal Analysis of IP 87.251.78.46 (6 detections, malicious behavior confirmed).

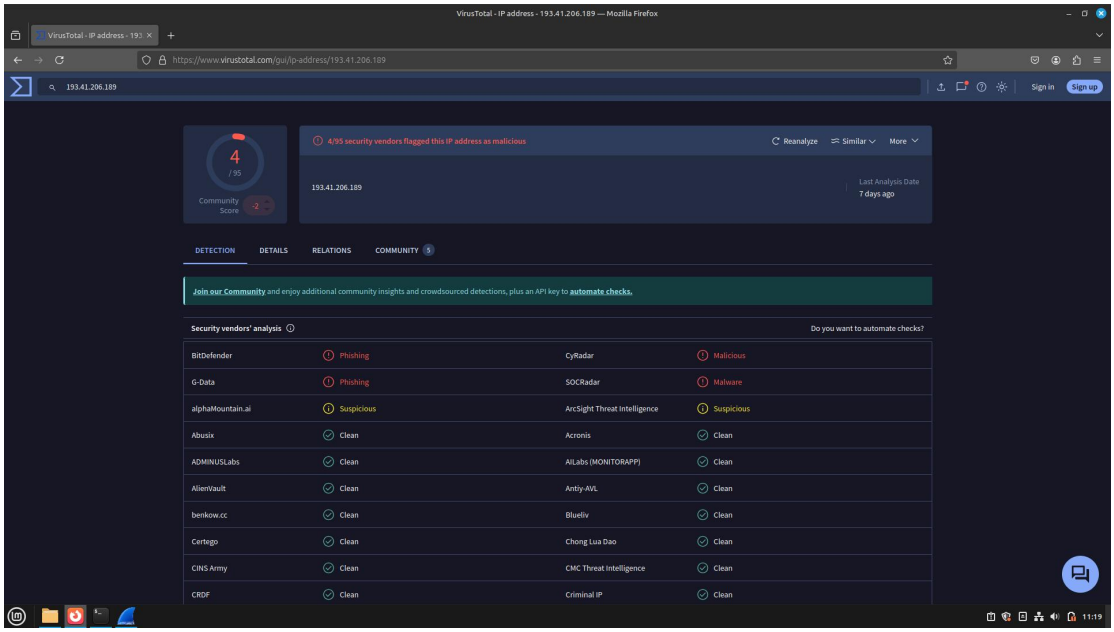

Fig A4: VirusTotal Analysis of IP 193.41.206.189 (4 detections, malicious behavior confirmed).

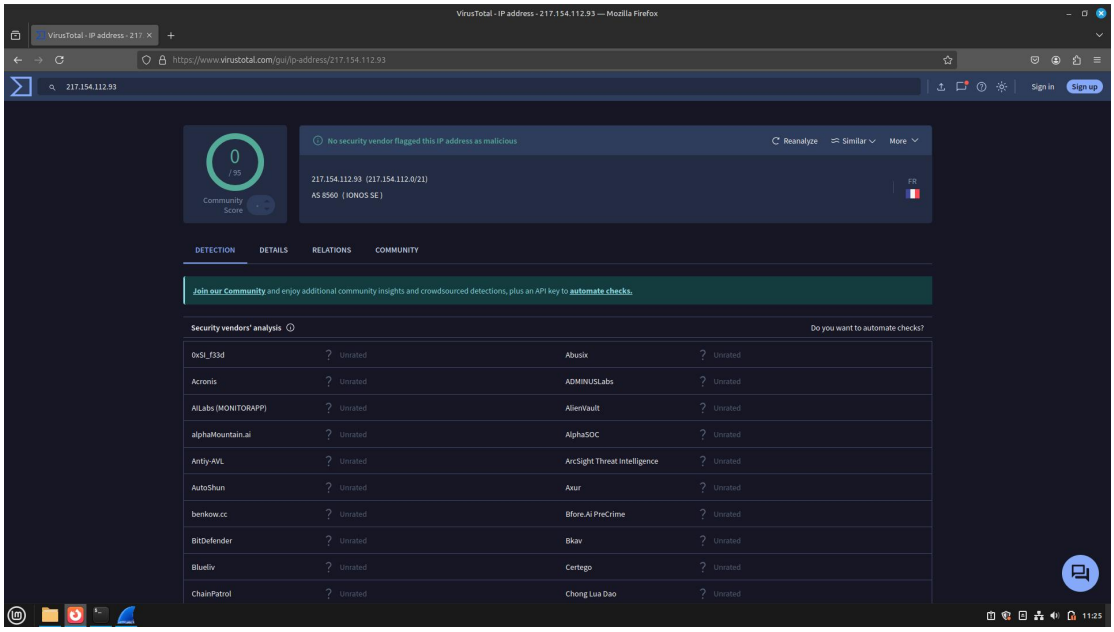

Fig A5: VirusTotal Analysis of IP 217.154.112.93 (0 detections, but suspicious Symfony probing behavior).