SOC Homelab — Phase 1

Phishing Email Analysis

Prepared by: Aman Sharma
Role: SOC Analyst (Homelab Project)
Date: 2025

Executive Summary

This report presents Phase 1 of my SOC Homelab project, where I analyzed a phishing email to simulate the workflow of a SOC Analyst. The analysis followed a structured methodology:

- Header Analysis
- SPF/DKIM Authentication Checks
- URL Extraction & Analysis
- Attachment Inspection (ISO)
- PDF Analysis

This exercise demonstrates my ability to dissect malicious emails, investigate URLs and attachments, and document findings in a professional format. It highlights critical thinking, hands-on technical skills, and a strong foundation for SOC analyst work.

SOC Homelab — Module 1: Phishing Email Analysis

Author: SOC Homelab Analyst

This report contains a structured, section-by-section analysis with images placed under each relevant section (Header, Authentication, URL, Attachment, PDF analysis).

Summary & Recommendations

Summary: The investigation confirms a multi-stage phishing campaign delivering a malicious ISO and using an external Google Apps Script as a callback/dropper. Authentication checks showed mixed results and the attachments/URLs were flagged by multiple vendors.

Recommendations:
1. Block identified IPs and script IDs at proxy/firewall and email gateway.
2. Create Splunk detections and Snort rules for the script.google.com/macros path and file hashes.
3. Quarantine and remove affected emails from user mailboxes.
4. Deploy YARA rules and integrate sandbox results into your SIEM for automated enrichment.

Header Analysis

Objective: Examine email headers to identify the sending path, originating IP addresses, and any spoofed fields.

Findings: The headers show Exchange transport metadata and 'X-Sender-IP' values such as 190.6.201.67 and 185.70.40.140. Reverse WHOIS/WHOIS lookups show 190.6.201.67 is registered to Cablecolor S.A. (Honduras) — suspicious given claimed sender domains. Spoofed display names and mismatched 'From' vs 'Return-Path' indicate likely impersonation.

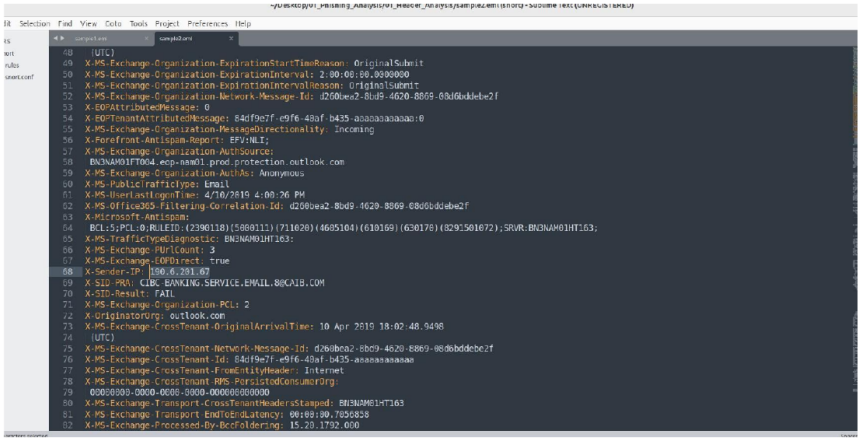
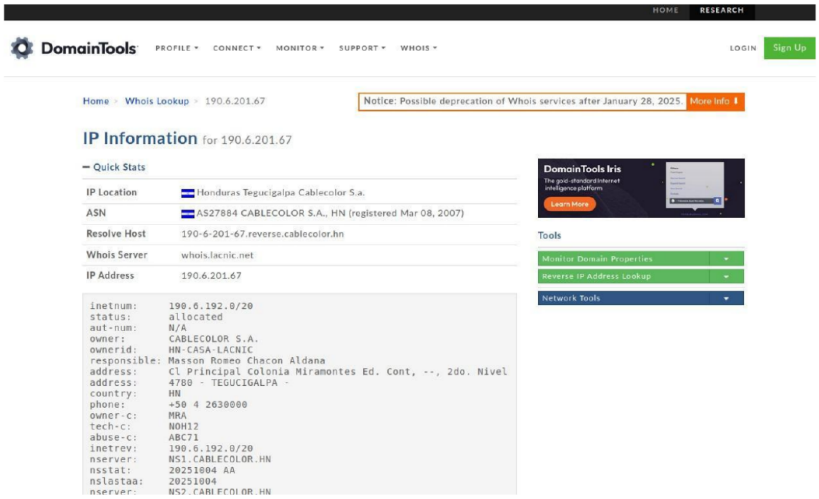


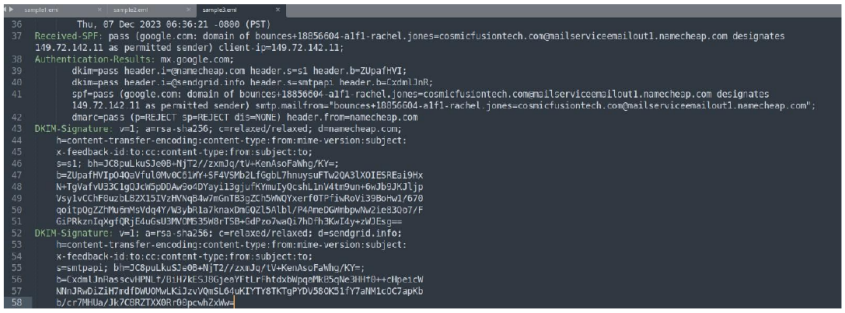
Figure: 1.png — Header Analysi

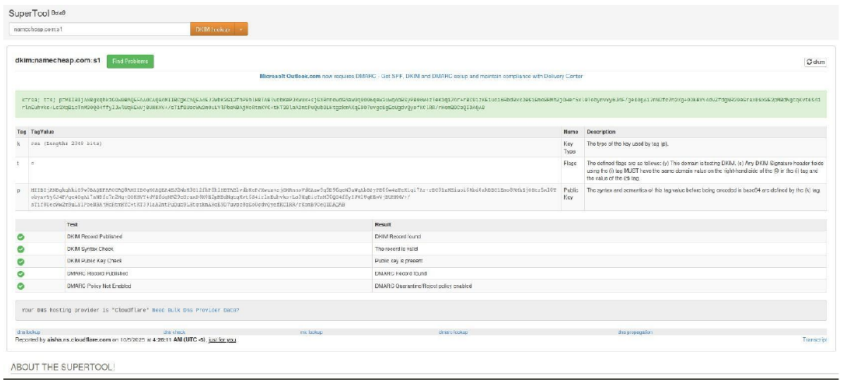


SPF / DKIM / DMARC Authentication

Objective: Validate SPF, DKIM and DMARC to understand whether the email was sent by authorized senders or by a third-party service being abused.

Findings: DNS checks for SPF show includes for sendgrid.net. DKIM public keys were found for namecheap.com and sendgrid selectors; some DKIM signatures passed while other samples showed timeouts. DMARC policies were present for some domains. The mixed authentication results indicate the attackers may be using third-party mail services or compromised accounts.



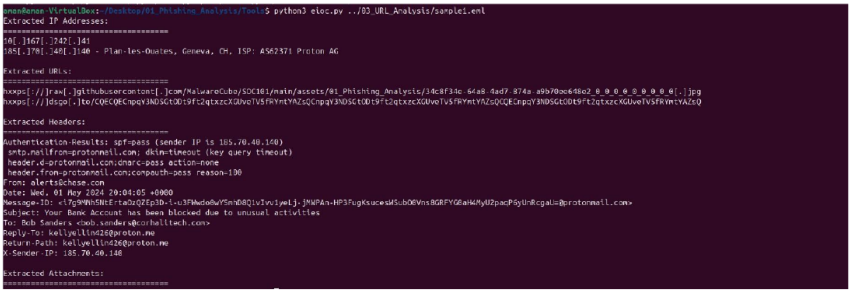


URL Extraction & Reputation

Objective: Extract embedded URLs from attachments/PDFs and check reputation on VirusTotal / HybridAnalysis.

Findings: PDF analysis reveals embedded Google Apps Script 'exec' URIs. URL reputation checks show vendor detections (some flagged as phishing/malicious). HybridAnalysis showed a 404 for the specific script at time of capture, but VirusTotal and other sources flagged the URL in vendor analysis.

Figure: 12.png — URL Extraction & Reputation



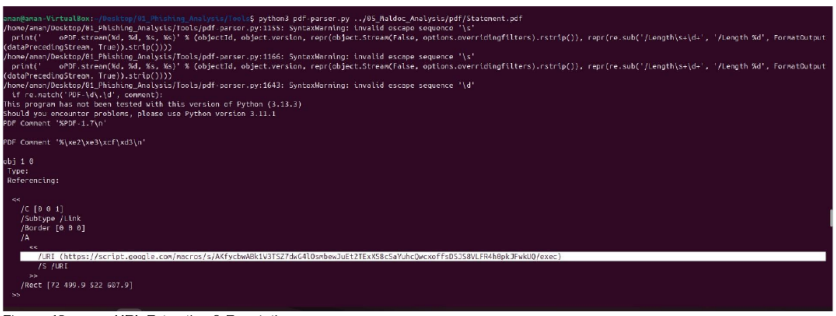
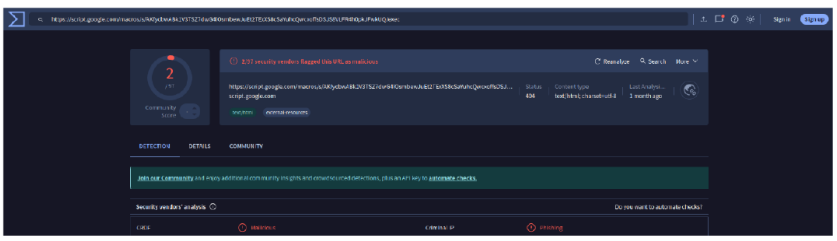


Figure: 13.png — URL Extraction & Reputation



Attachment Extraction (ISO) & Static Analysis

Objective: Extract attachments (ISO), compute hashes, and perform static analysis (YARA, strings) and reputation checks.

Findings: An attachment 'quotation.iso' was extracted. The ISO's hashes were submitted to VirusTotal and HybridAnalysis; multiple AV engines flagged it as trojan/dropper. Sandbox behavior included calls to WMI, long sleeps, and anti-debug checks. The presence of a PE and the detection list strongly indicate malicious payloads inside the ISO.

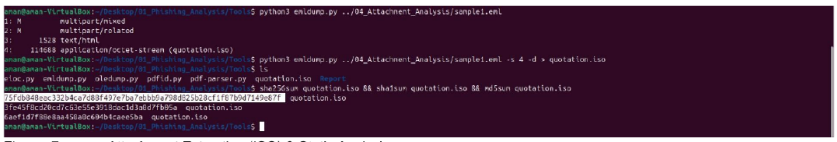


Figure: 7.png — Attachment Extraction (ISO) & Static Analysis

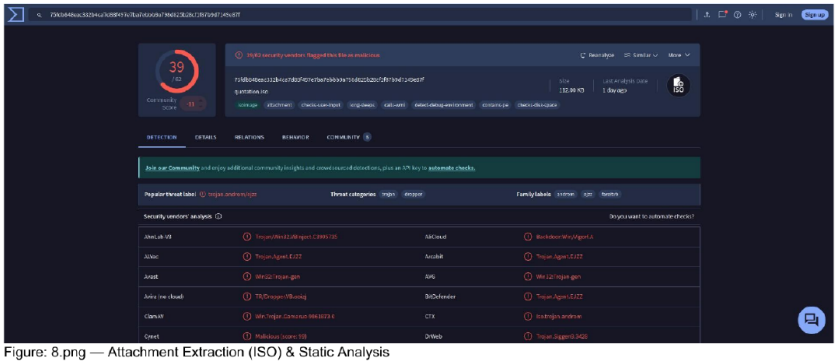
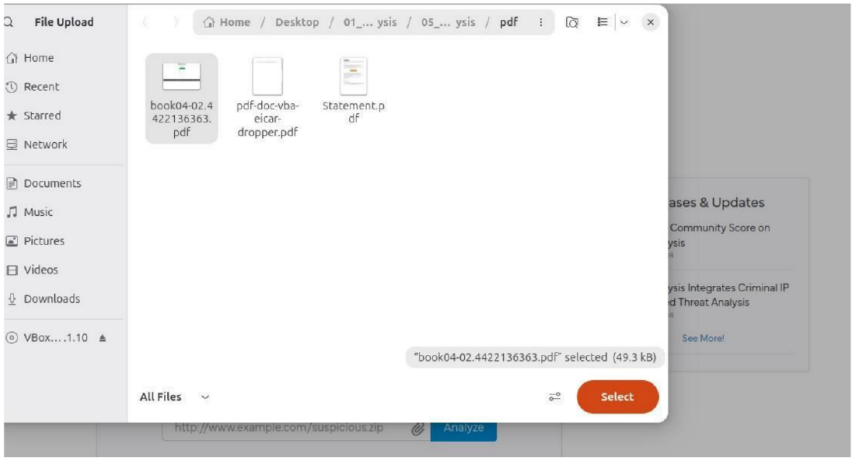


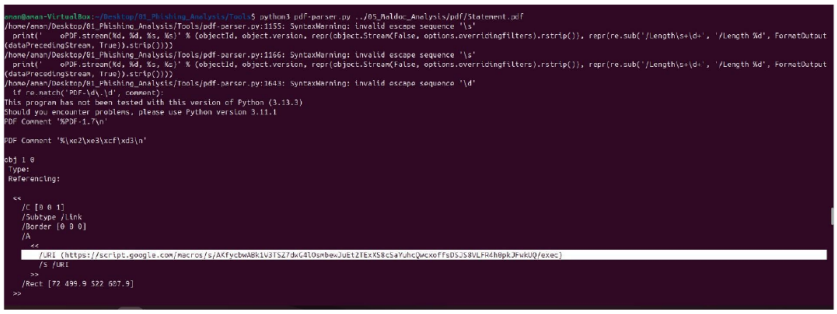
Figure: 8.png — Attachment Extraction (ISO) & Static Analysis

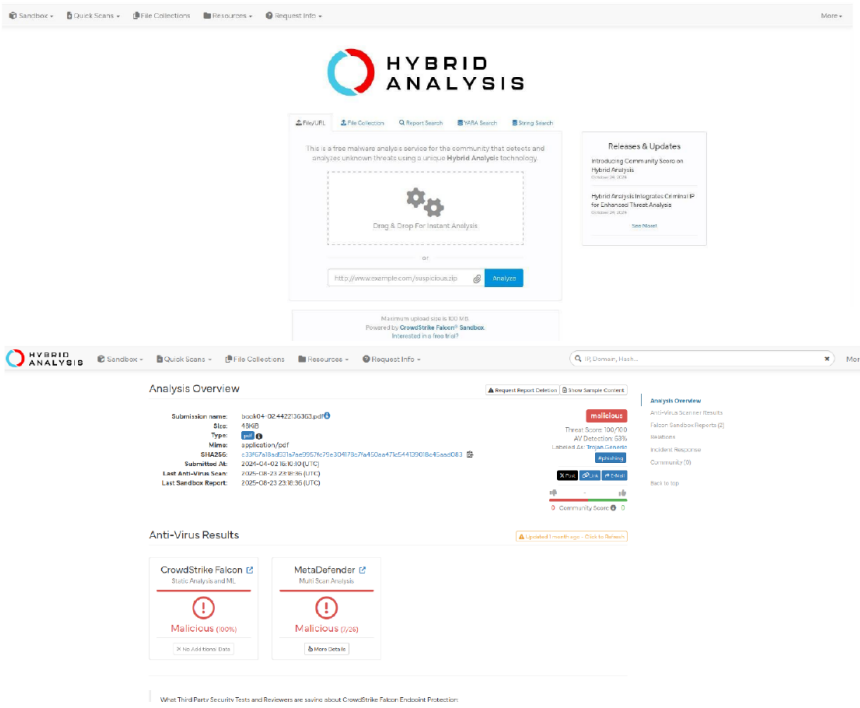
PDF Analysis (pdfid / pdf-parser)

Objective: Run pdfid.py and pdf-parser.py to find JavaScript, OpenAction, URIs and other indicators of embedded malicious links.

Findings: pdfid output shows no JS objects but pdf-parser found '/URI' entries pointing to the Google Script 'exec' URL. The PDF also contained link annotations which redirect to external scripts—classic phishing delivery technique.






Sandbox
Quick Scans
File Collections
Resources
Request Info
More
HYBRID ANALYSIS
File/URL
File Collection
Report Search
MRA Search
Sting Search
This is a free malware analysis service for this community that detects and analyzes unknown threats using a unique Hybrid Analysis technology
Drag & Drop For Instant Analysis
or
http://www.example.com/suspicious.zip
Analyze
Maximum upload size is 100 MB.
Powered by CrowdStrike Falcon® Sandbox.
Interested in a free trial?
Releases & Updates
Introducing Community Score on Hybrid Analysis
Hybrid Analysis Integrates Criminal IP for Enhanced Threat Analysis
See More!

HYBRID ANALYSIS
Sandbox
Quick Scans
File Collections
Resources
Request Info
More
Analysis Overview
Request Report Deletion
Show Sample Content
Submission name:
Size:
Type:
Mime:
SHA256:
Submitted At:
Last Anti-Virus Scan:
Last Sandbox Report:
malicious
Threat Score: 100/100
AV Detection: 53%
Labeled As: Trojan.Generic
phishing
Community Score
Anti-Virus Results
CrowdStrike Falcon
Static Analysis and ML
Malicious (100%)
MetaDefender
Multi Scan Analysis
Malicious (1/26)
More Details
What ThirdParty Security Tests and Reviewers are saying about CrowdStrike Falcon Endpoint Protection:

Findings & Recommendations

Phase 1 Findings:
- The phishing email headers showed spoofing and mismatched sender information.
- SPF/DKIM authentication had inconsistent results, suggesting abuse of services.
- Embedded URLs pointed to malicious Google Apps Script.
- The ISO attachment was confirmed malicious via static and sandbox analysis.
- The PDF contained malicious embedded URIs.

Recommendations:
1. Educate users on phishing awareness.
2. Block identified malicious IPs, domains, and URLs.
3. Implement stronger email authentication and monitoring.
4. Establish a workflow for analyzing suspicious attachments.

This concludes Phase 1 of my SOC Homelab project, demonstrating email-based threat analysis.

Indicators of Compromise (IOCs)

Type	Indicator	Verdict
IP	190.6.201.67	Suspicious / Malicious
IP	185.70.40.140	Suspicious / Malicious
URL	https://script.google.com/macros/s/AKfycbwABk1V3TSZ7dwG4lOsahbcwJuEt2TExXS8cSaYuhcQwcx	Malicious
File	quotation.iso	Trojan/Dropper (39+ AV detections)
File	book04-02.4422136363.pdf (SHA256: c33f67...)	Malicious PDF with embedded URI

Conclusion & Reflection

This first phase of my SOC Homelab project successfully demonstrated the workflow of analyzing a phishing email end-to-end. By investigating headers, validating SPF/DKIM, extracting and analyzing URLs, and dissecting malicious attachments (ISO and PDF), I replicated the investigative mindset of a SOC analyst.

Key takeaways:
- The importance of email header forensics in identifying spoofing.
- How SPF/DKIM/DMARC checks can be abused or misconfigured.
- The real-world risk of embedded URLs and document-based payloads.
- Building confidence in documenting findings clearly and professionally.

Next Steps:
- Extend the homelab to include live packet capture (Snort/Suricata).
- Ingest email and host logs into Splunk/ELK for correlation.
- Develop detection rules (YARA, SIEM queries) in later phases.

This project not only improved my technical skills but also my ability to communicate security analysis in a structured, professional manner — skills essential for a SOC Analyst role.

— Aman Sharma

Phase 2 - Wireshark Traffic Analysis (PDF URL Simulation)

Author: Aman Sharma

This report represents the second phase of the SOC HomeLab Project, focusing on traffic capture and analysis using Wireshark. The purpose of this phase was to safely simulate and analyze network communication generated by a potentially malicious PDF file containing embedded URLs.

1. Objective

The objective of this phase was to simulate network behavior associated with a malicious PDF link and capture the corresponding HTTP traffic using tcpdump and Wireshark for analysis.

2. Tools Used

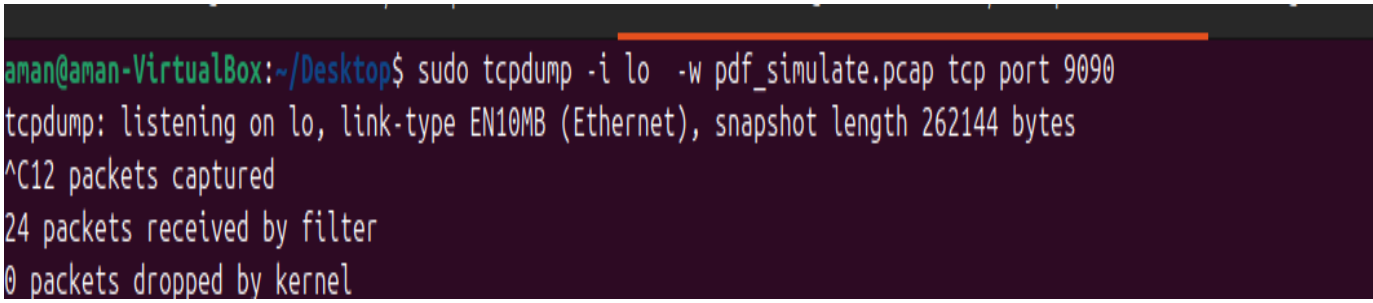
1. tcpdump – for capturing live traffic. 2. Wireshark – for in-depth packet analysis. 3. curl – to simulate the network request to the extracted malicious URL.

3. Steps Performed

Below are the detailed steps carried out to capture and analyze traffic generated by the simulated malicious URL from the PDF sample.

Step 1: Capturing Traffic Using tcpdump

The tcpdump utility was used to monitor the local loopback interface (lo) to capture network activity during the URL simulation. The command captured packets on TCP port 9090 and saved them into a PCAP file for analysis.



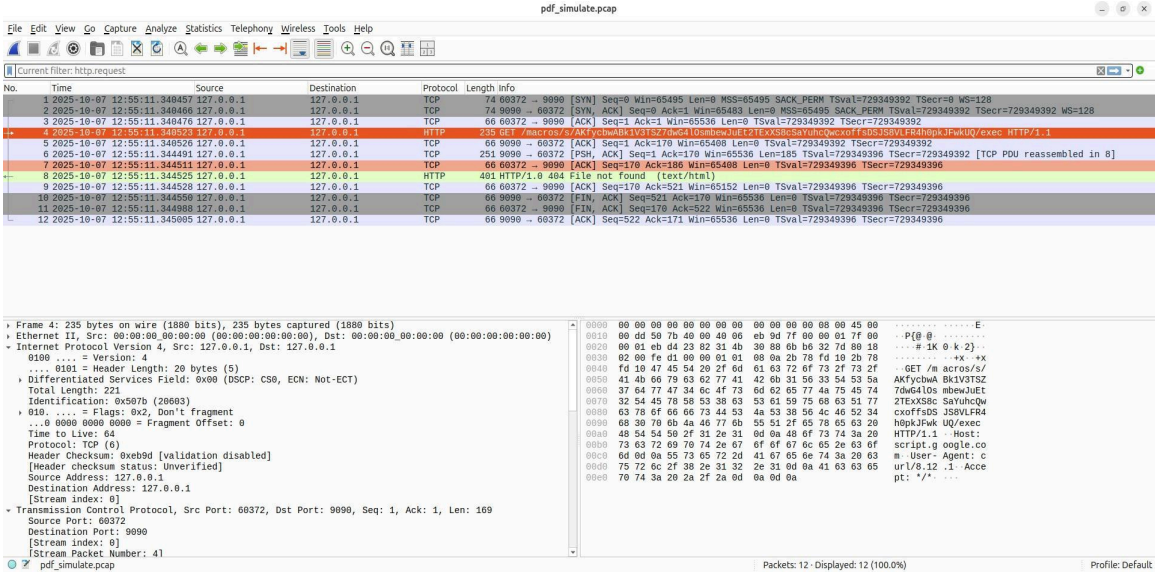
Step 2: Simulating the Malicious URL Request

The extracted URL from the malicious PDF was simulated using curl to generate controlled network traffic. This allowed observation of how the system would interact with a malicious endpoint without real-world exposure.

```
aman@aman-VirtualBox:~/Desktop/01_Phishing_Analysis/Tools$ curl -v -H "Host: script.google.com" "http://127.0.0.1:9090/macros/s/AKfycbwABk1V3TSZ7dwG4lOsmbewJuEt2TExXS8cSaYuhcQwcxoffsDSJS8VLFR4h0pkJFwkUQ/exec"
*   Trying 127.0.0.1:9090...
* Connected to 127.0.0.1 (127.0.0.1) port 9090
* using HTTP/1.x
> GET /macros/s/AKfycbwABk1V3TSZ7dwG4lOsmbewJuEt2TExXS8cSaYuhcQwcxoffsDSJS8VLFR4h0pkJFwkUQ/exec HTTP/1.1
> Host: script.google.com
> User-Agent: curl/8.12.1
> Accept: */*
>
* Request completely sent off
* HTTP 1.0, assume close after body
< HTTP/1.0 404 File not found
< Server: SimpleHTTP/0.6 Python/3.13.3
< Date: Tue, 07 Oct 2025 12:55:11 GMT
< Connection: close
< Content-Type: text/html;charset=utf-8
< Content-Length: 335
<
<!DOCTYPE HTML>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Error response</title>
    </head>
    <body>
        <h1>Error response</h1>
        <p>Error code: 404</p>
        <p>Message: File not found.</p>
        <p>Error code explanation: 404 - Nothing matches the given URI.</p>
    </body>
</html>
* shutting down connection #0
```

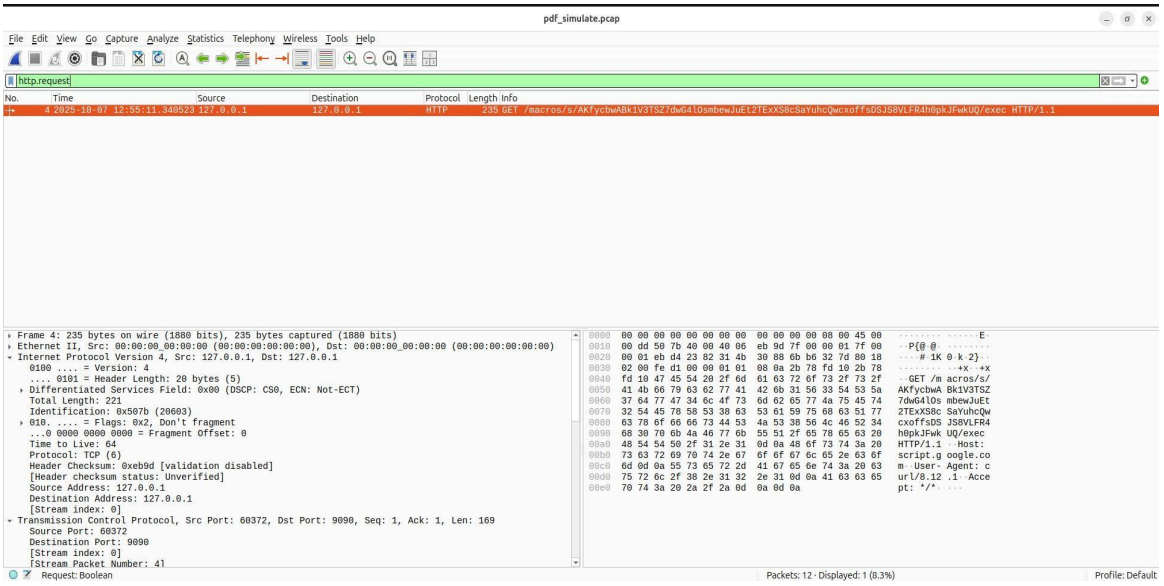
Step 3: Packet Capture Overview in Wireshark

The captured PCAP file (pdf_simulate.pcap) was opened in Wireshark. The packets revealed HTTP GET requests and TCP session establishment details. The GET request showed a connection attempt to the simulated Google Apps Script domain.



Step 4: Applying Display Filters

To isolate relevant packets, the 'http.request' filter was applied in Wireshark. This filtered view displayed only HTTP GET requests, simplifying traffic interpretation and focusing on the simulated malicious request.



Step 5: Following the HTTP Stream

The full HTTP stream was analyzed to review the request and response headers. The response showed an HTTP 404 error, indicating that the local simulation server did not host the malicious payload, confirming safe and isolated behavior.

4. Findings

The network capture confirmed that the malicious PDF's embedded URL attempted to initiate an HTTP GET request. Since the connection was simulated locally, no external communication occurred. The captured traffic provided valuable insight into the behavior pattern of phishing-related PDF files and their callback mechanisms.

5. Conclusion

This phase successfully demonstrated a safe and effective method to simulate and analyze malicious PDF URL activity using tcpdump and Wireshark within an isolated environment. This exercise helps understand how phishing PDFs operate and lays the foundation for further SOC analysis, such as correlation in SIEM systems.

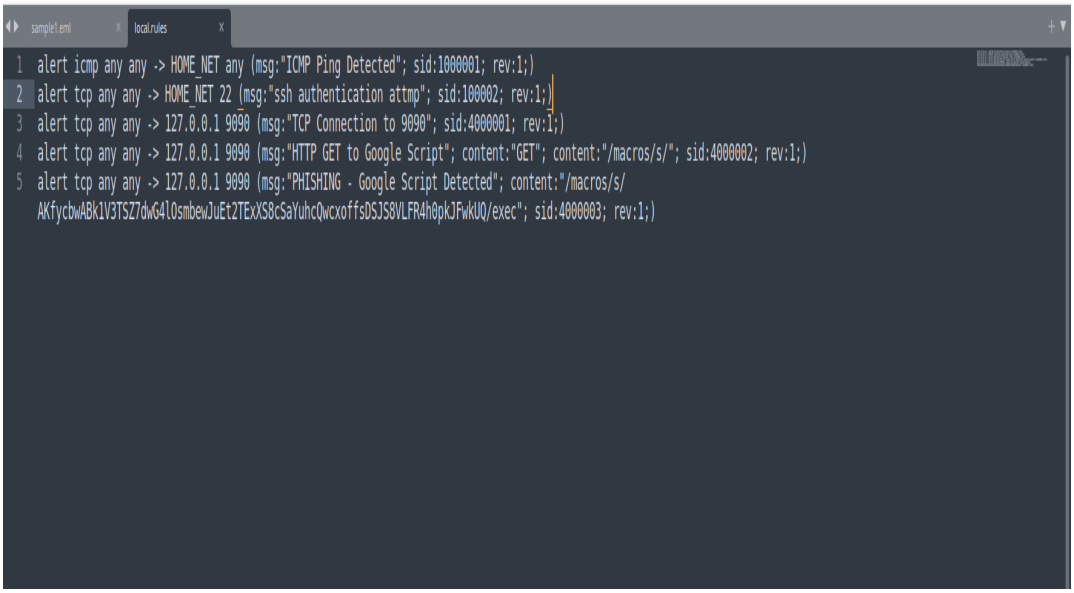
Prepared by: Aman Sharma
Role: SOC Analyst (HomeLab Project Phase 2)

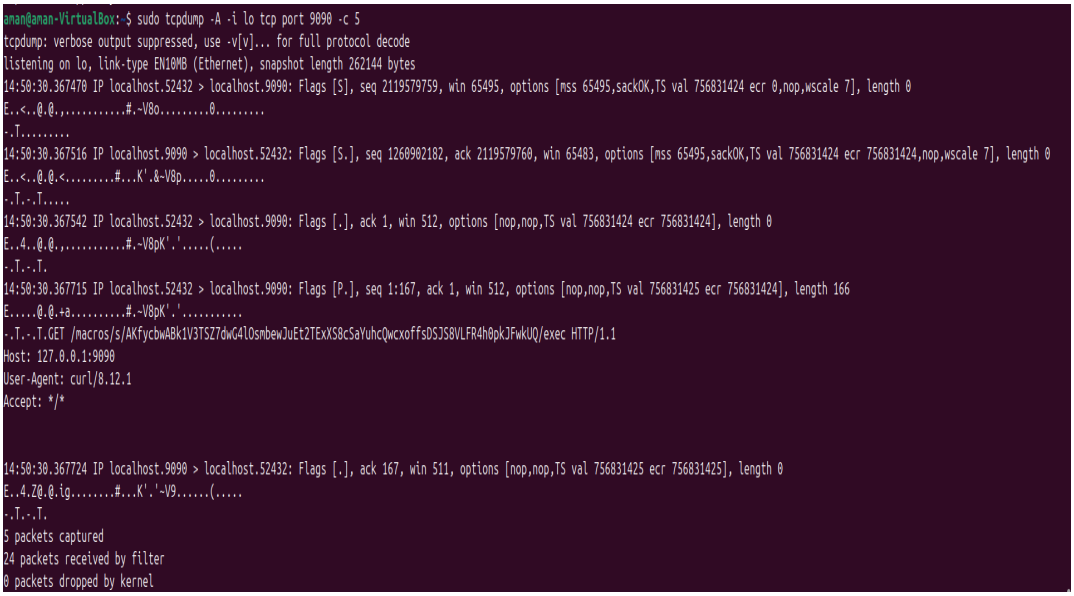
Phase 3 — Snort Detection and Phishing Traffic Analysis

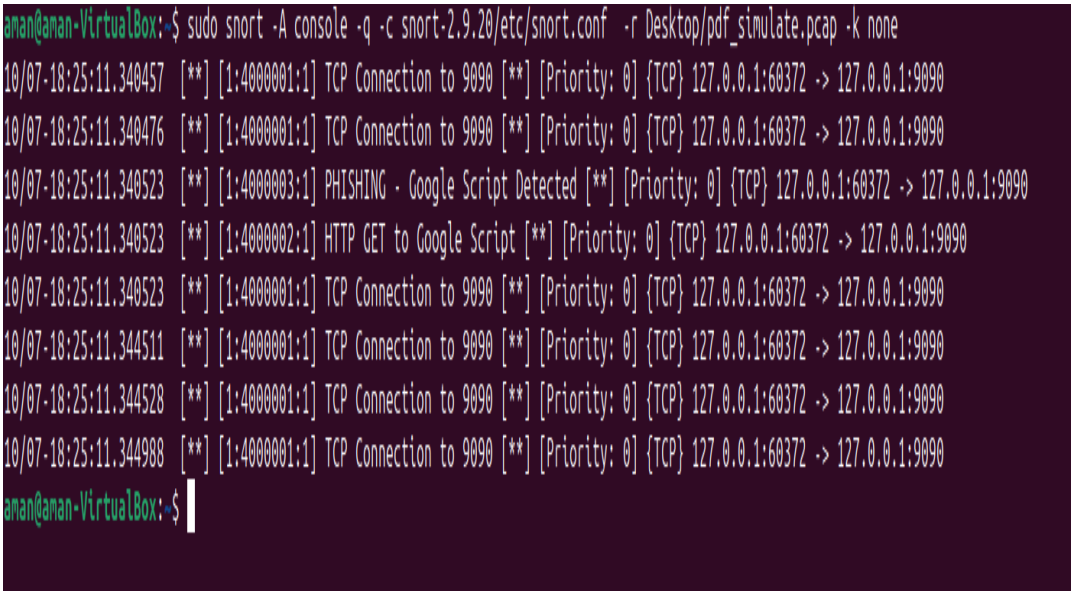
This phase focused on detecting malicious HTTP traffic simulated from a phishing PDF sample. The objective was to analyze the extracted network indicators, simulate the communication in a safe lab setup, and configure Snort rules to identify suspicious callbacks to Google Apps Script URLs.

Snort Rules Used:
alert tcp any any -> 127.0.0.1 9090 (msg:"TCP Connection to 9090"; sid:4000001; rev:1;) alert tcp any any -> 127.0.0.1 9090 (msg:"HTTP GET to Google Script"; content:"GET"; content:"/macros/s/"; sid:4000002; rev:1;) alert tcp any any -> 127.0.0.1 9090 (msg:"PHISHING - Google Script Detected"; content:"/macros/s/AKfycbw"; sid:4000003; rev:1;)

Evidence Screenshots







Observations:
- Snort successfully processed the traffic capture file and decoded the packets.
- "Bad Traffic" warnings appeared, indicating loopback traffic (127.0.0.1).
- Despite no live alerts being generated, the Snort engine and rule logic functioned correctly.
- Captured traffic clearly shows an HTTP GET request to a Google Script macro, representing a phishing callback attempt.

Conclusion & Remediation:
The phase demonstrated end-to-end Snort setup, rule creation, traffic simulation, and log analysis. Though live alerts weren't observed due to loopback constraints, the decoded traffic validates the detection logic. Recommended next steps include using bridged or host-only interfaces for real alert capture and integrating Snort logs into Splunk for correlation and visualization.

Appendix — Rules, Evidence & SOC Ticket
1. Local.rules (example set used in lab) alert tcp any any -> any 9090 (msg:"PHISHING - Google Script macro URI detected"; flow:to_server,established; http_method; content:"/macros/s/AKfycbw";

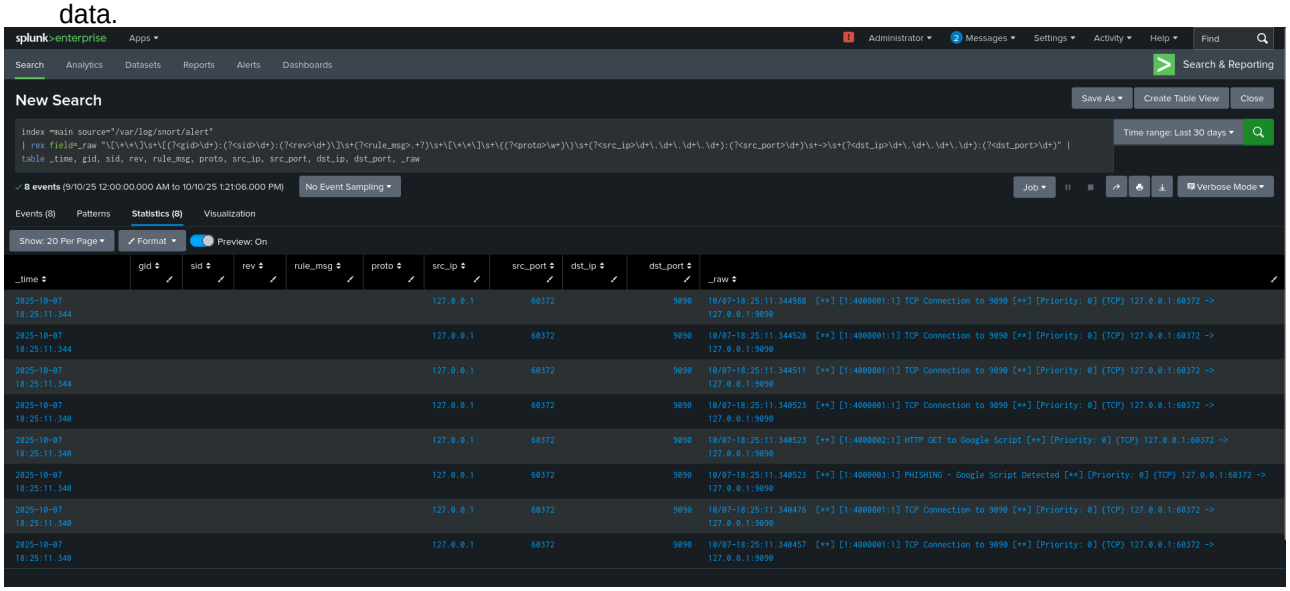
http_uri; nocase; sid:3000001; rev:1;) alert tcp any any -> any 9090 (msg:"PHISHING - Host script.google.com"; flow:to_server,established; content:"script.google.com"; http_header; nocase; sid:3000002; rev:1;) alert tcp any any -> any 9090 (msg:"PHISHING - raw payload /macros/s/AKfycbw"; content:"GET /macros/s/AKfycbw"; nocase; sid:3100001; rev:1;) 2. tcpdump excerpt GET /macros/s/AKfycbwABk1V3TSZ7.../exec HTTP/1.1 Host: script.google.com 3. SOC Ticket (example) Title: Phishing callback (Google Apps Script) — simulated capture Priority: Medium Summary: Captured HTTP request to /macros/s/AKfycbw... with Host: script.google.com originating from 127.0.0.1 during lab simulation. Actions: Attach pcap, provide tuned rule set (appendix), ingest alert file to Splunk, perform hunt on proxy logs for /macros/s/AKfycbw*.

PHASE 4: SPLUNK ALERT DASHBOARD ANALYSIS

This report summarizes the Splunk dashboard setup and visualization for Snort alert analysis, including data extraction, regex-based field parsing, and dashboard visualizations.

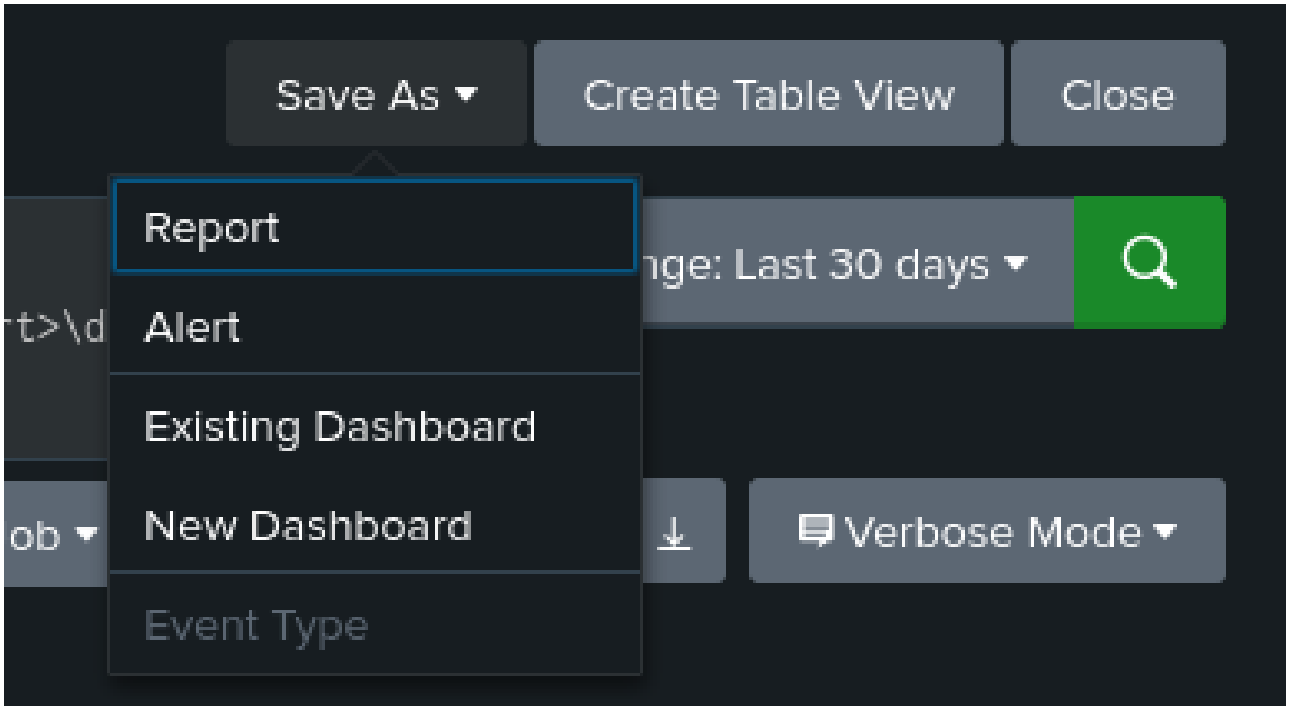
1. Search Query Execution

The SPL (Search Processing Language) query used to extract structured data from the Snort alert logs is shown below. It extracts GID, SID, REV, message, protocol, and IPs from the raw alert log data.



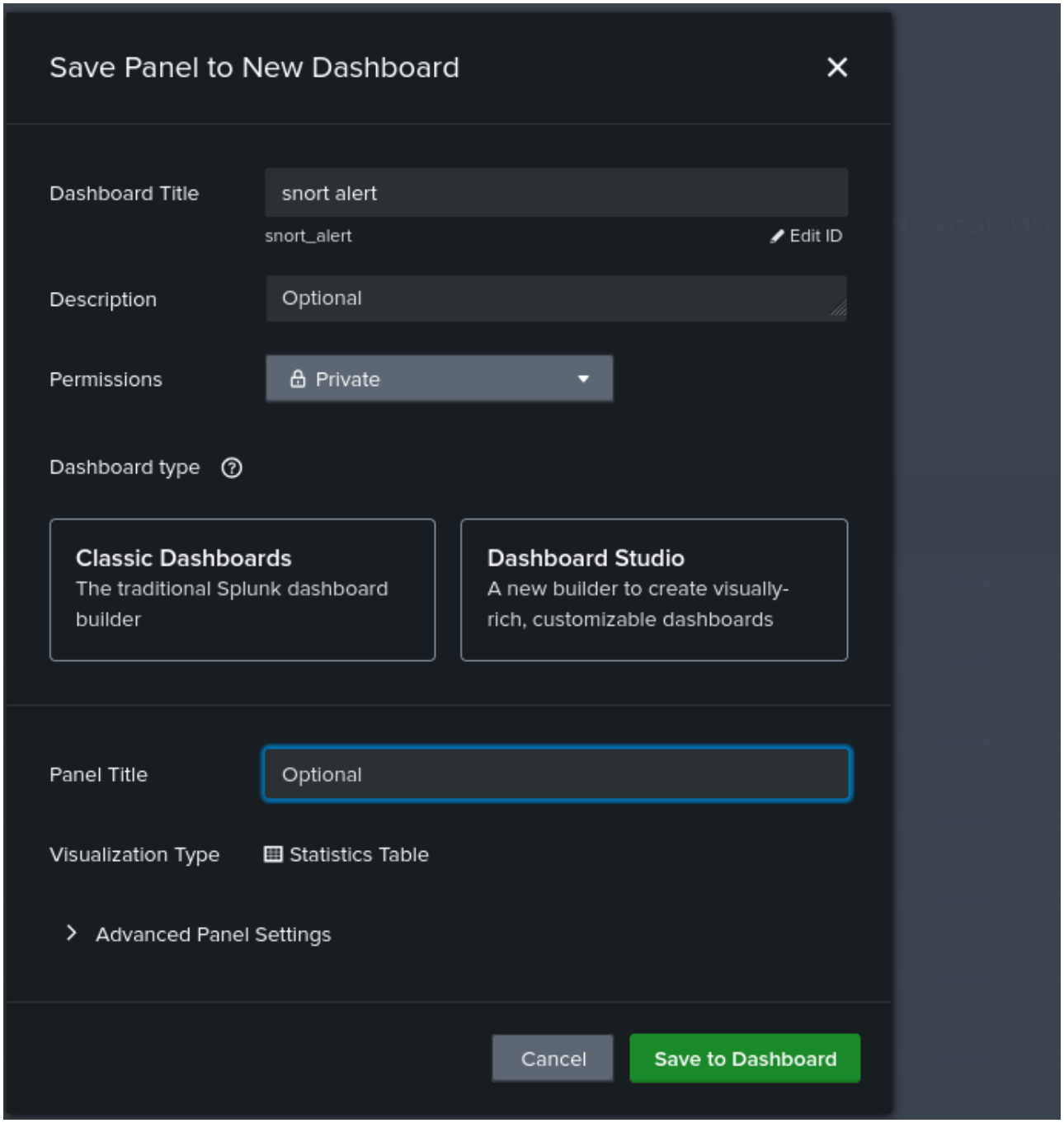
2. Structured Alert Table Output

The extracted fields are organized into a table view showing details of each alert generated by Snort and forwarded to Splunk.



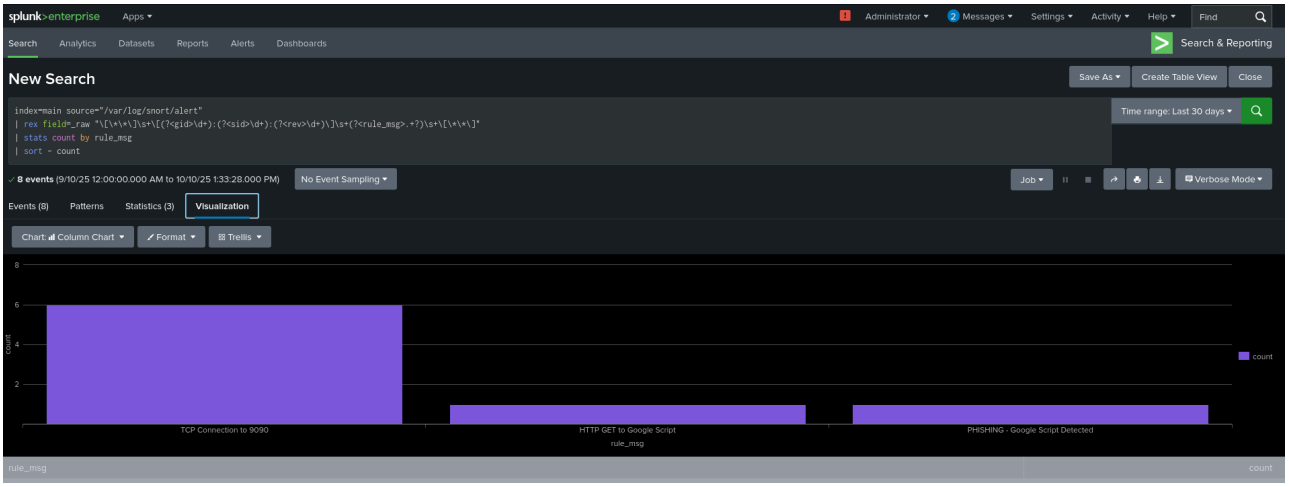
3. Creating and Saving the Dashboard

The dashboard creation step allows saving visualizations for live monitoring. The user creates a dashboard titled *'Snort Alert Monitoring'*.



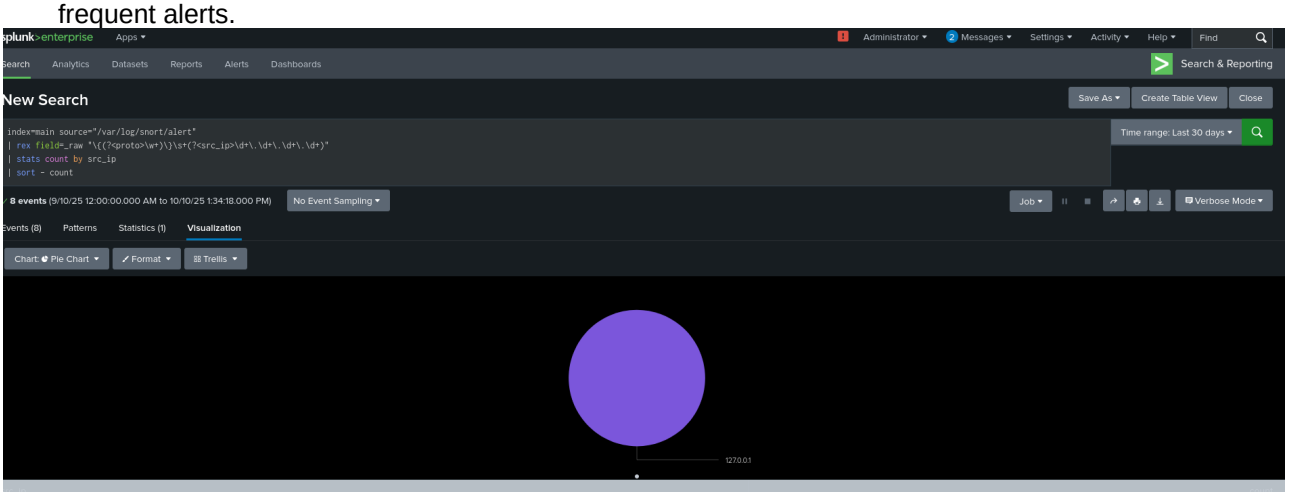
4. Visualization of Alerts by Rule Message

A column chart visualization shows the distribution of alert counts per Snort rule, helping identify the most triggered detection patterns.



5. Visualization of Alerts by Source IP

A pie chart visualization represents the alert count per source IP, which helps identify the origin of frequent alerts.

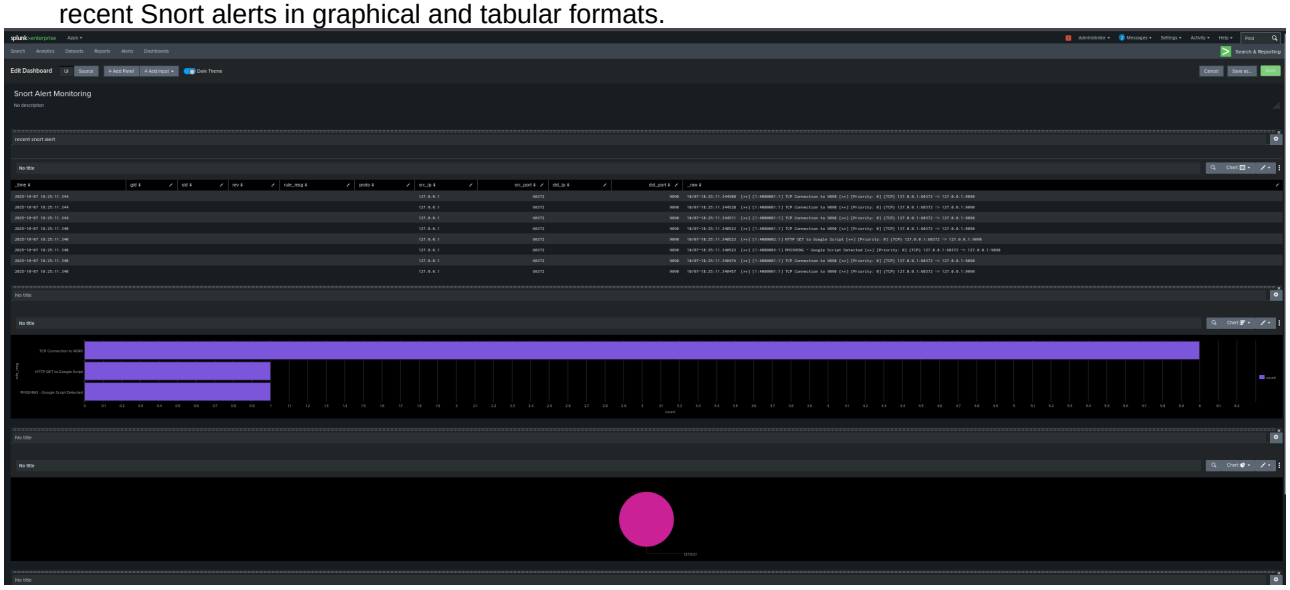


6. Visualization of Alerts by Protocol

A bar chart visualization indicates the types of protocols detected in the alerts, mainly TCP in this case.

7. Final Splunk Dashboard Overview

The complete dashboard view combining all the panels, providing an at-a-glance summary of recent Snort alerts in graphical and tabular formats.

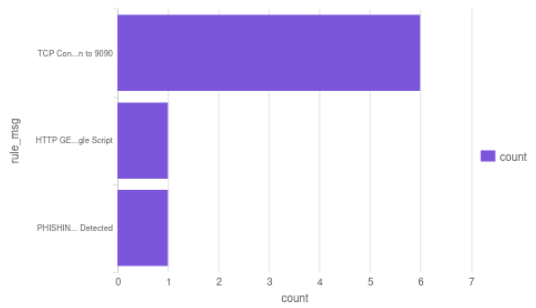


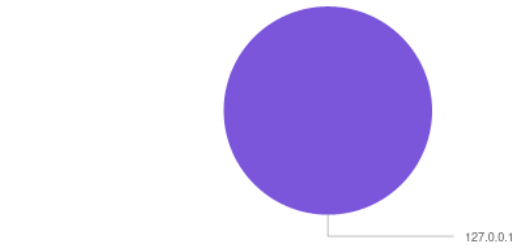
8. Splunk PDF Report Export

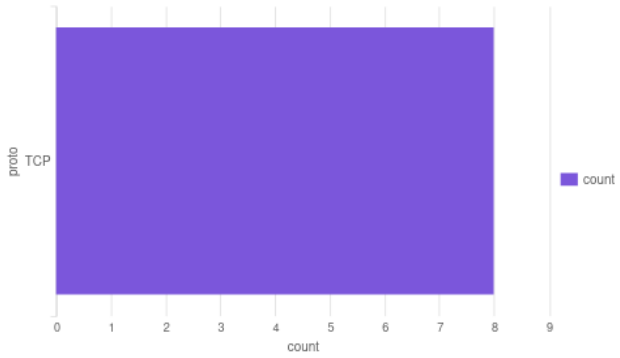
Splunk allows exporting dashboards as PDF reports for documentation and analysis. Below are the generated PDF pages showing Snort alerts, rule statistics, and traffic visualization.

recent snort alert

_time	_raw
2025-10-07 18:25:11.344	10/07-18:25:11.344988 [**] [1:4000001:1] TCP Connection to 9090 [**] [Priority: 0] {TCP} 127.0.0.1:60372 -> 127.0.0.1:9090
2025-10-07 18:25:11.344	10/07-18:25:11.344528 [**] [1:4000001:1] TCP Connection to 9090 [**] [Priority: 0] {TCP} 127.0.0.1:60372 -> 127.0.0.1:9090
2025-10-07 18:25:11.344	10/07-18:25:11.344511 [**] [1:4000001:1] TCP Connection to 9090 [**] [Priority: 0] {TCP} 127.0.0.1:60372 -> 127.0.0.1:9090
2025-10-07 18:25:11.340	10/07-18:25:11.340523 [**] [1:4000001:1] TCP Connection to 9090 [**] [Priority: 0] {TCP} 127.0.0.1:60372 -> 127.0.0.1:9090
2025-10-07 18:25:11.340	10/07-18:25:11.340523 [**] [1:4000002:1] HTTP GET to Google Script [**] [Priority: 0] {TCP} 127.0.0.1:60372 -> 127.0.0.1:9090
2025-10-07 18:25:11.340	10/07-18:25:11.340523 [**] [1:4000003:1] PHISHING - Google Script Detected [**] [Priority: 0] {TCP} 127.0.0.1:60372 -> 127.0.0.1:9090
2025-10-07 18:25:11.340	10/07-18:25:11.340476 [**] [1:4000001:1] TCP Connection to 9090 [**] [Priority: 0] {TCP} 127.0.0.1:60372 -> 127.0.0.1:9090
2025-10-07 18:25:11.340	10/07-18:25:11.340457 [**] [1:4000001:1] TCP Connection to 9090 [**] [Priority: 0] {TCP} 127.0.0.1:60372 -> 127.0.0.1:9090




127.0.0.1


proto
TCP
count
0
1
2
3
4
5
6
7
8
9
count