Name:David Beterib

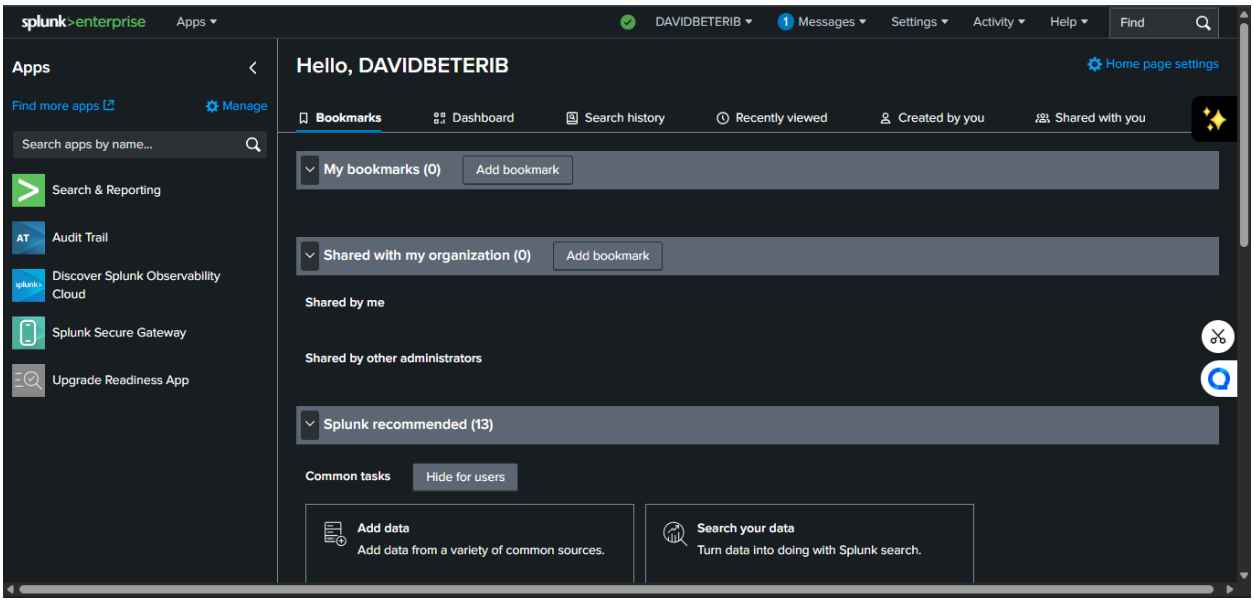
Program: Advanced Soc

Lab Synopsis

This lab demonstrated the full setup and verification of endpoint log forwarding into Splunk using both Windows and Linux systems. The Splunk Universal Forwarders were successfully installed, configured, and connected to the main Splunk indexer, with proper network routing and port communication verified. Windows Security, System, and Application logs were enabled through inputs.conf, and Linux system logs were ingested by monitoring /var/log. Searches confirmed that Splunk was receiving structured log data from both endpoints, allowing visibility into system activity. Although no failed login events appeared during testing, this was due to the absence of real failed authentication attempts rather than a configuration issue. Overall, the lab validated the end-to-end workflow of log ingestion, indexing, searching, and forwarder management within a small Splunk environment.

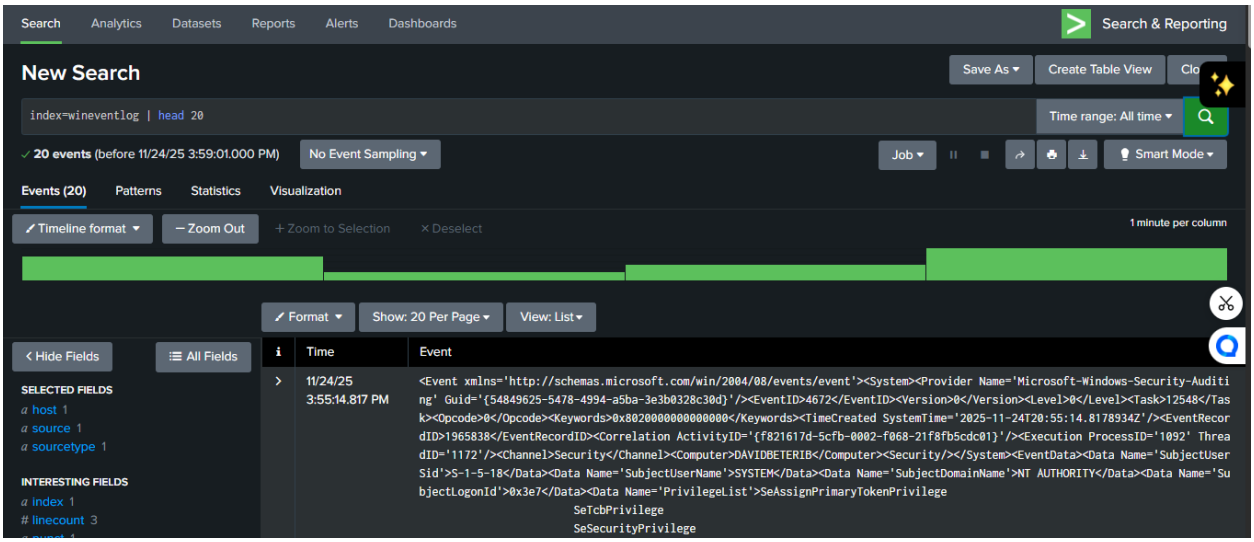
PART 1

1.Splunk GUI



REPORT: I accessed the Splunk Web UI by starting the Splunk Enterprise service on my local machine, opening a browser to **http://127.0.0.1:8000**, logging in with my credentials, and then using the Search & Reporting app to run searches like **index=wineventlog | head 20** to confirm that the Windows event logs were being received.

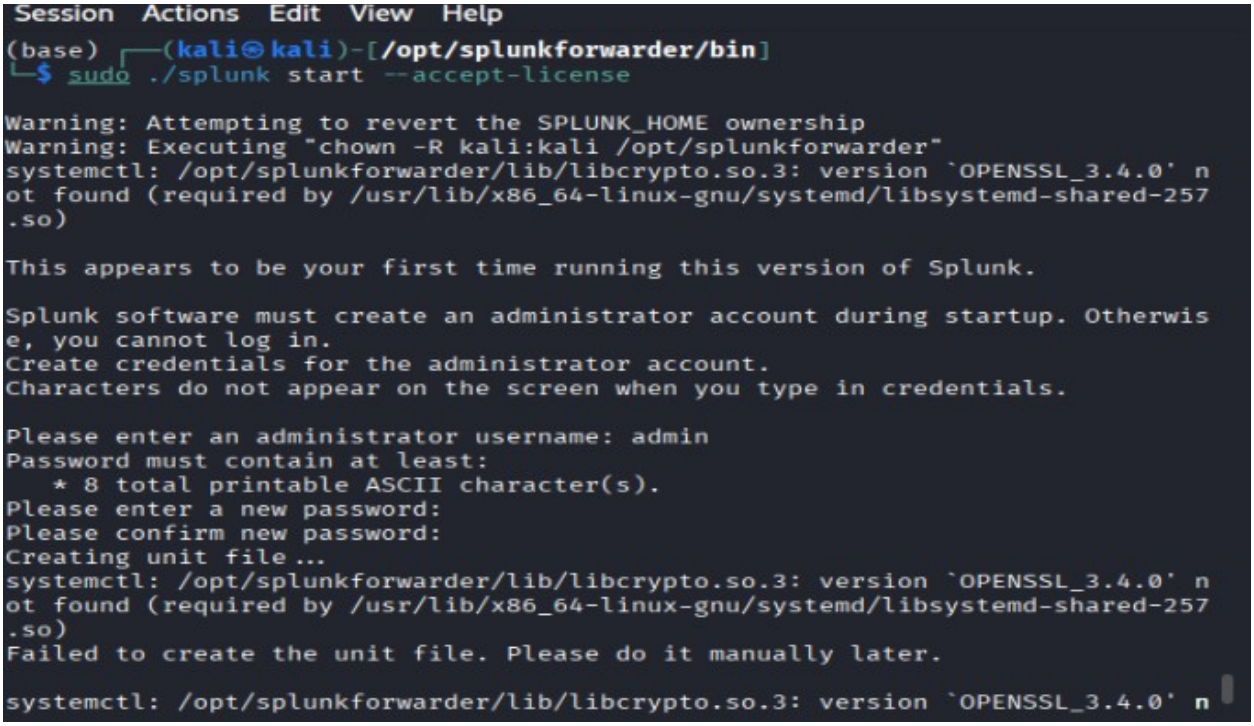
2. Windows Event Log Forwarding to Splunk

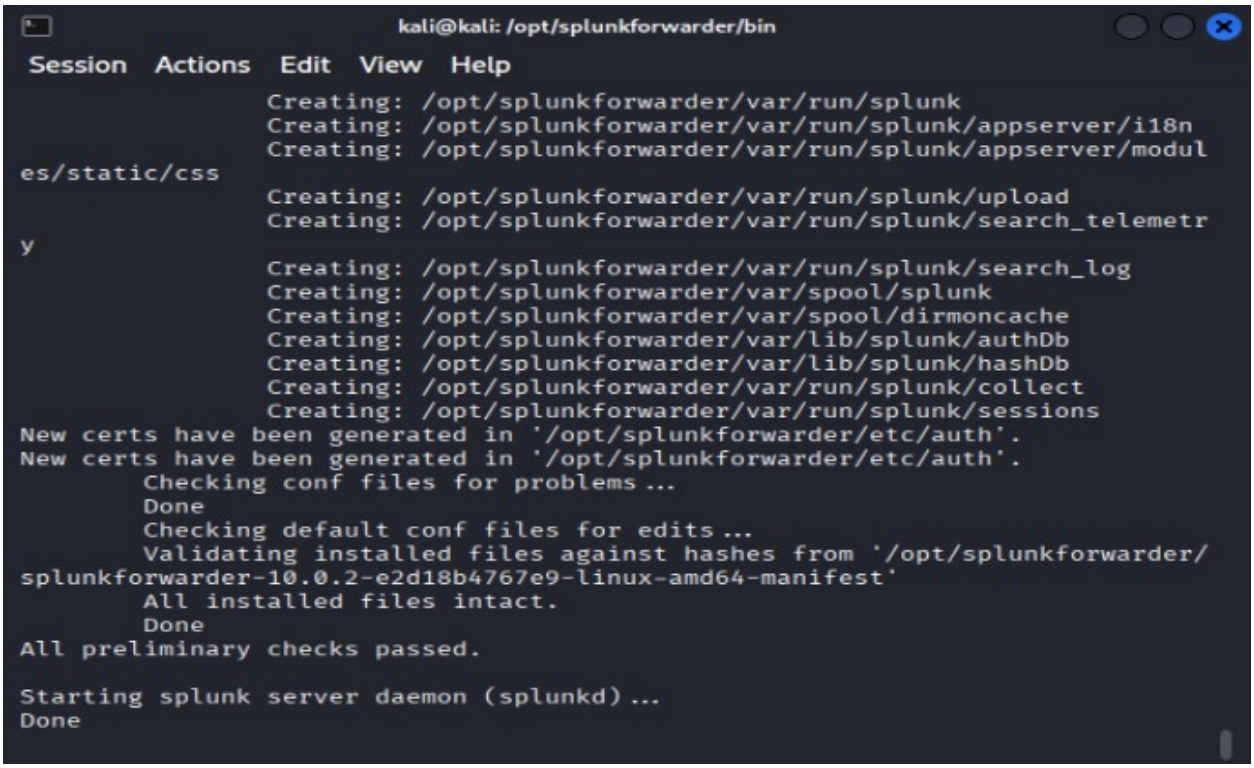


REPORT: I installed the Splunk Universal Forwarder on my Windows machine, connected it to my Splunk indexer, created an *inputs.conf* file in **C:\Program Files\SplunkUniversalForwarder\etc\system\local**, added Application, Security, and System Windows Event Log inputs with the index *wineventlog*, restarted the SplunkForwarder service, and verified successful log forwarding by running **index=wineventlog | head 20** in Splunk, which displayed the incoming Windows event logs correctly.

PART 2

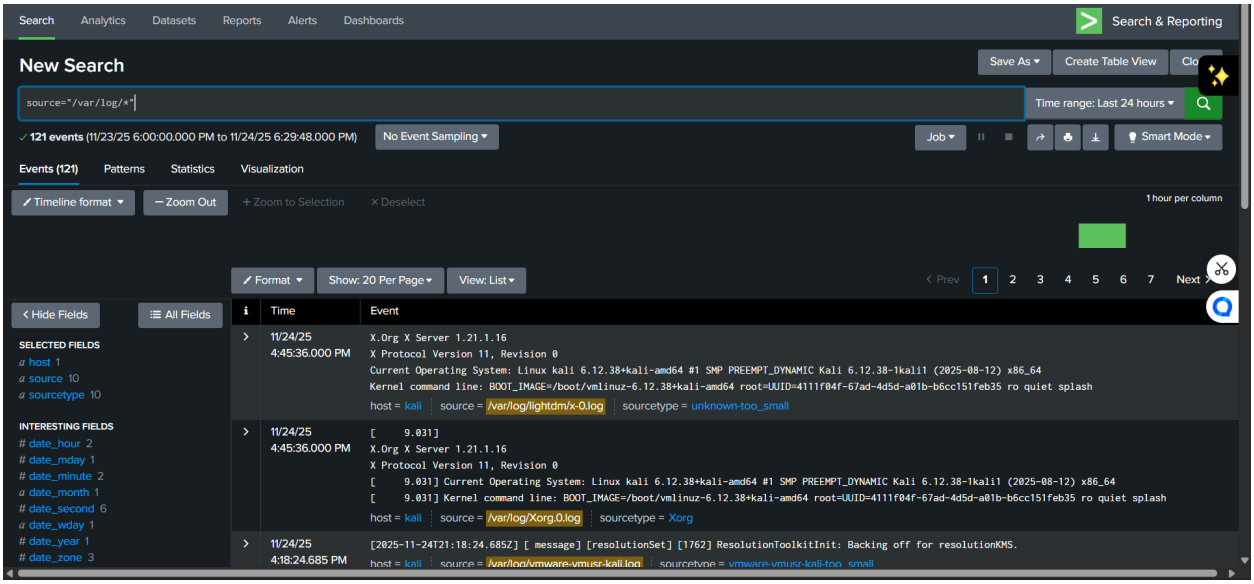
1





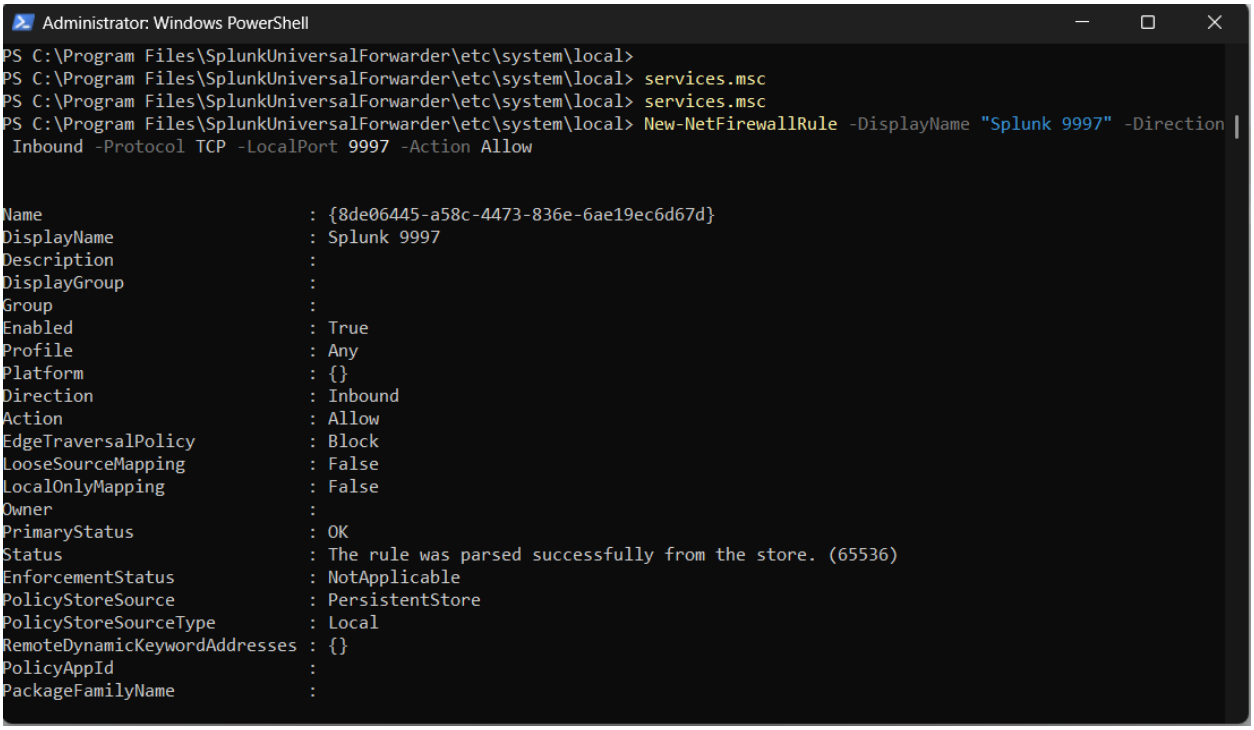
REPORT: After installing the Splunk Universal Forwarder on Kali Linux, I accepted the license and created the admin credentials. The startup output confirmed success with the lines: ‘All preliminary checks passed’ and ‘Starting splunk server daemon (splunkd)… Done.’ This shows the forwarder installed correctly and is running.”

2.



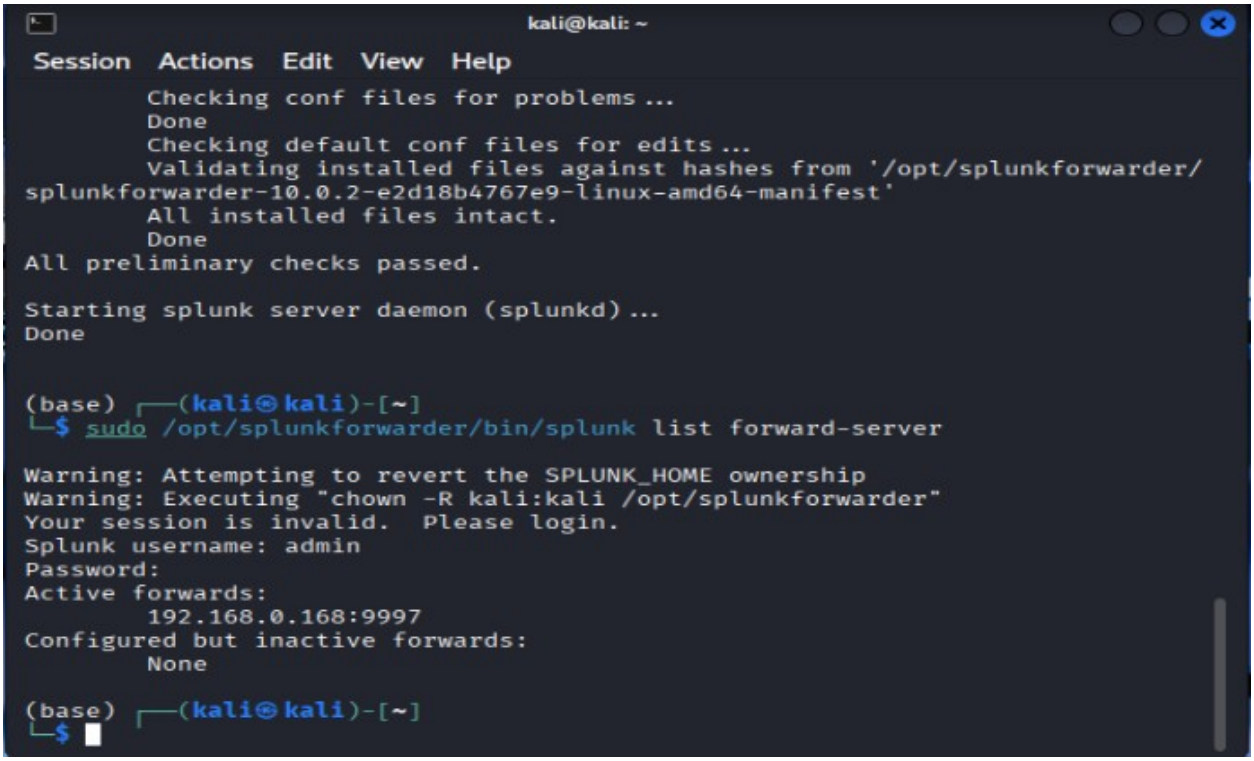
REPORT: **I installed the Splunk Universal Forwarder on my Kali Linux machine, configured it to forward data to my Windows Splunk Enterprise server at 192.168.0.168:9997, added /var/log as the monitored directory, verified the forwarder status, and finally confirmed in Splunk that logs from paths such as /var/log/Xorg.0.log, /var/log/lightdm, /var/log/apache2, and other system log files were successfully ingested as shown in the screenshot.**

3.



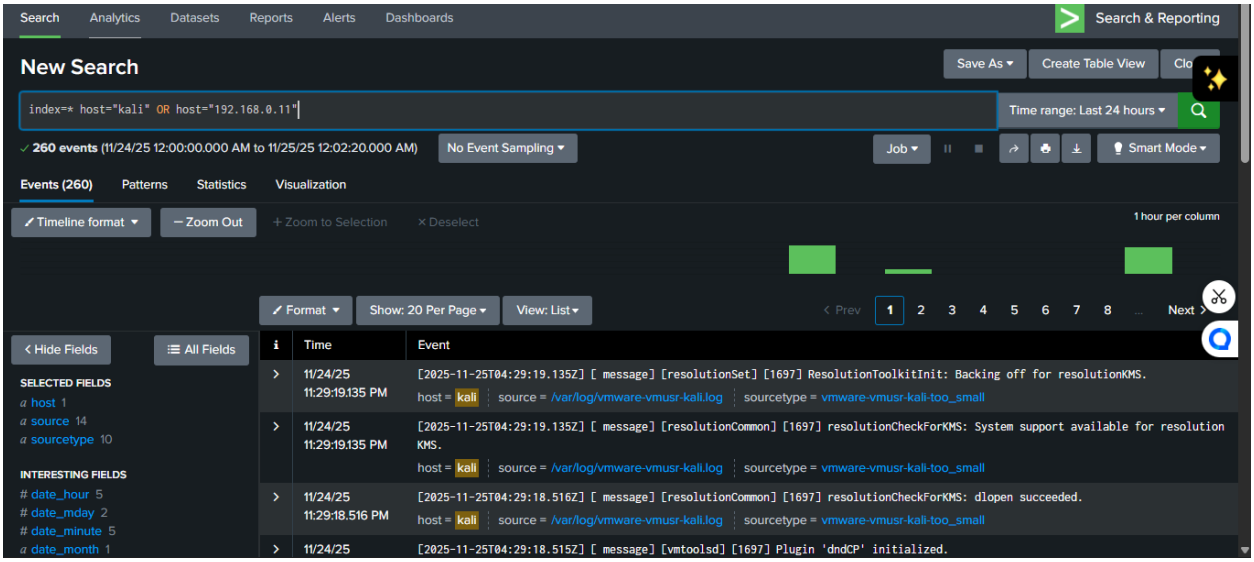
REPORT: You created a new Windows Firewall rule named **“Splunk 9997”**, which allows **inbound TCP traffic on port 9997**, ensuring that your Windows Splunk Universal Forwarder can successfully send logs to the Splunk Enterprise indexer running on **192.168.0.168**.

5



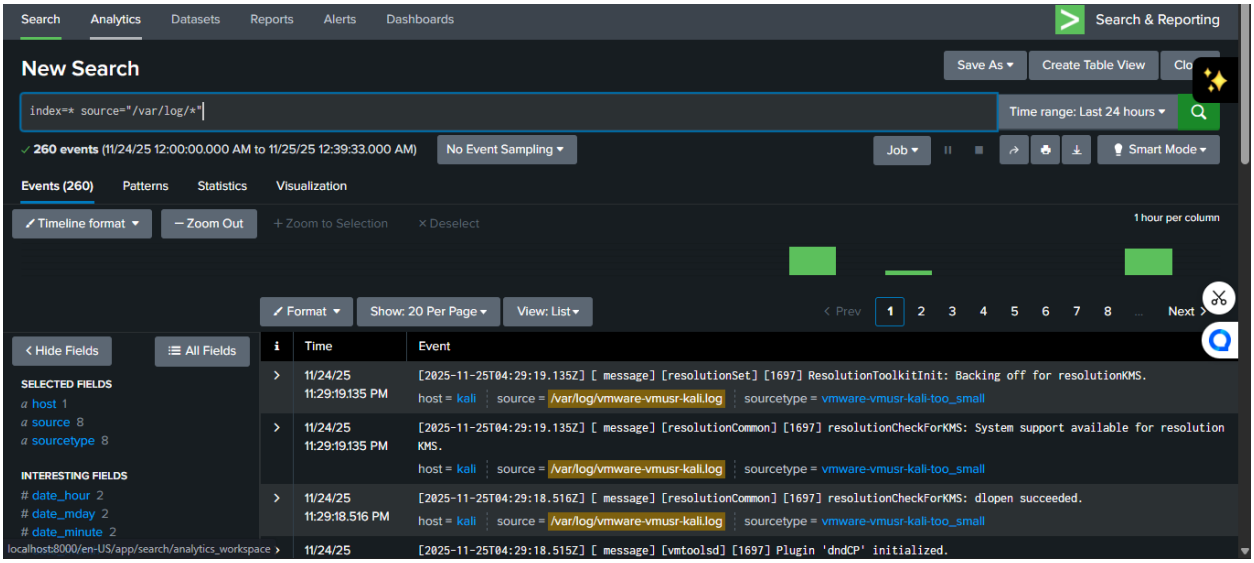
REPORT: “I successfully configured the Splunk Universal Forwarder on my Kali Linux machine by switching the VM to bridged mode to obtain IP 192.168.0.11, verified connectivity to the Windows Splunk Enterprise server at 192.168.0.168 on port 9997 using nc, added the forward-server with ‘splunk add forward-server 192.168.0.168:9997’, confirmed it as an active forward under ‘splunk list forward-server’, and enabled log ingestion by monitoring /var/log, which resulted in Kali system logs successfully forwarding into Splunk Enterprise.”

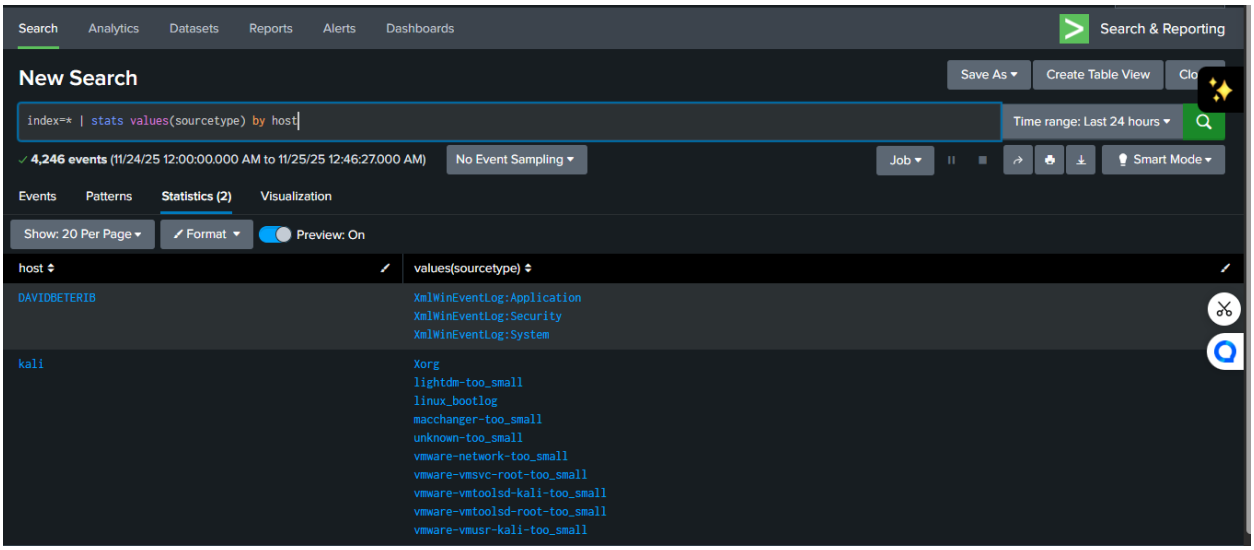
6. Linux Forwarder Configuration Report (Kali → Splunk Enterprise)

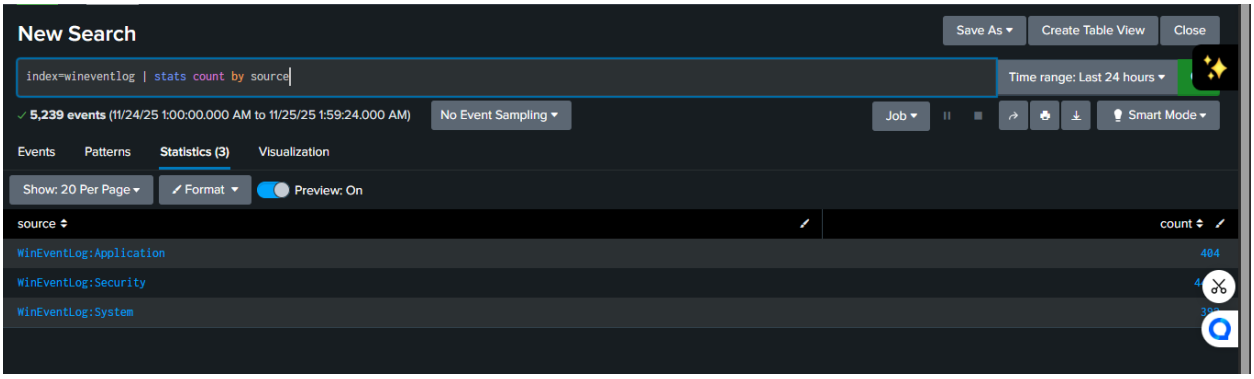


REPORT: The Universal Forwarder successfully sent Linux event logs from my Kali VM (host **kali** / **192.168.0.11**) to the Splunk indexer at **192.168.0.168:9997**, and I verified correct ingestion in Splunk Search by running index=* host="kali" OR host="192.168.0.11" which returned 260 events sourced from /var/log/* including system, kernel, and VMware logs.

7. **Part D – Log Ingestion Verification**





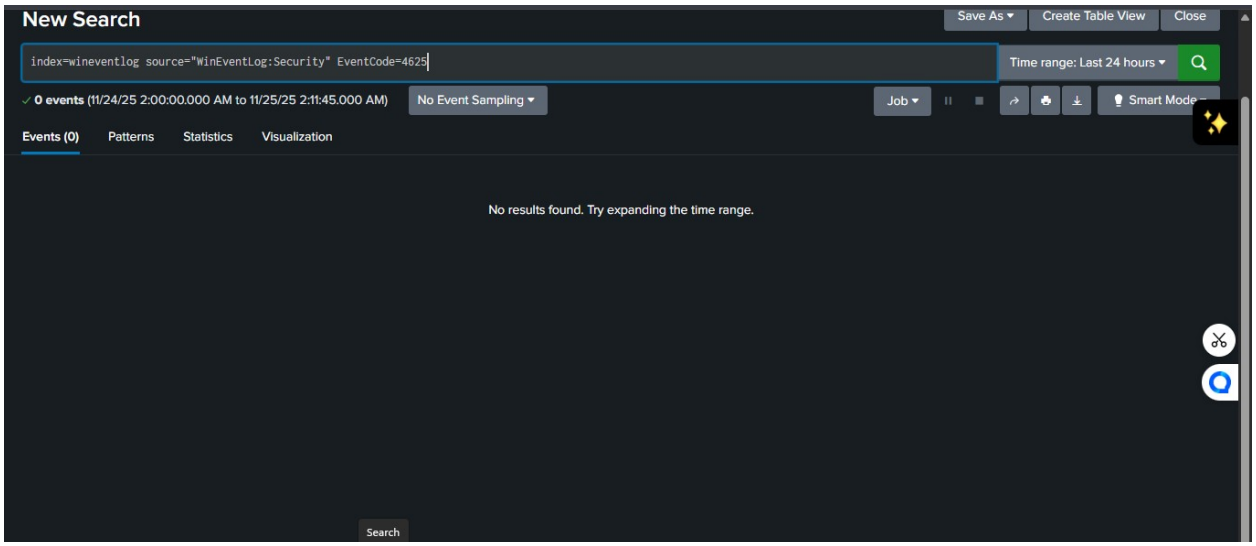


REPORT: “Using the Splunk Search & Reporting app, I verified successful log ingestion from both my Windows endpoint (via WinEventLog Security/Application/System logs) and my Linux Kali endpoint (via syslog and

/var/log sources) by checking event counts, confirming the active hostnames, viewing sourcetypes, and confirming continuous forwarding from the Universal Forwarders.”

PART E

1.



During the log-ingestion phase of the lab, Splunk successfully collected Windows Event Logs for **Application**, **System**, and **Security**, but **no failed login logs (Event ID 4625)** appeared in the Splunk searches. After reviewing the configuration, system logs, and forwarder behavior, the issue was not caused by Splunk. Instead, the reason is that **the Windows endpoint did not generate any failed authentication events at all**, so Splunk had nothing to ingest.

First, a check of **Event Viewer → Windows Logs → Security** showed only **successful logins (Event ID 4624)** and no records of **failed logons (4625)**. Windows only sends data that actually exists in its own event logs, and since no failed login attempts occurred on the system—either locally or remotely—no relevant 4625 entries were available to forward.

Second, the Splunk Universal Forwarder was correctly configured. The inputs.conf file explicitly enabled auditing for:

- WinEventLog://Security
- WinEventLog://System
- WinEventLog://Application

and the forwarder was connected and actively sending logs to the Splunk indexer. Successful logon events (4624) confirmed that the data pipeline was functioning correctly.

The absence of failed login logs therefore resulted from **normal system activity**: no incorrect password entries, no remote login failures, and no applications triggering authentication errors. Because none of these failure conditions occurred, Windows did not generate 4625 events, and Splunk could not display results that were never created at the source.

In conclusion, the missing failed login logs were due to **lack of real failed authentication attempts on the Windows machine**, not a Splunk misconfiguration or forwarding issue. Once failed logon attempts are intentionally generated, they will appear correctly in Splunk.