Comprehensive Analysis of Zeek Tunnel Traffic in Splunk SIEM

Objectives

- Validate successful ingestion and indexing of Zeek tunnel logs within Splunk.
- Conduct initial exploratory analysis to confirm data visibility and field extraction accuracy.
- Extract and quantify dominant tunnel protocols and their statuses to identify prevalent tunneling methods.
- Visualize temporal patterns in tunnel protocol activity to detect spikes or anomalies.
- Map tunnel traffic by source and destination IP addresses, correlating protocols and traffic volume for security insights.
- Highlight top source IPs by tunnel activity to pinpoint potential threat actors or misconfigurations.

1. Initial Inspection of Zeek Tunnel Traffic via Splunk SIEM

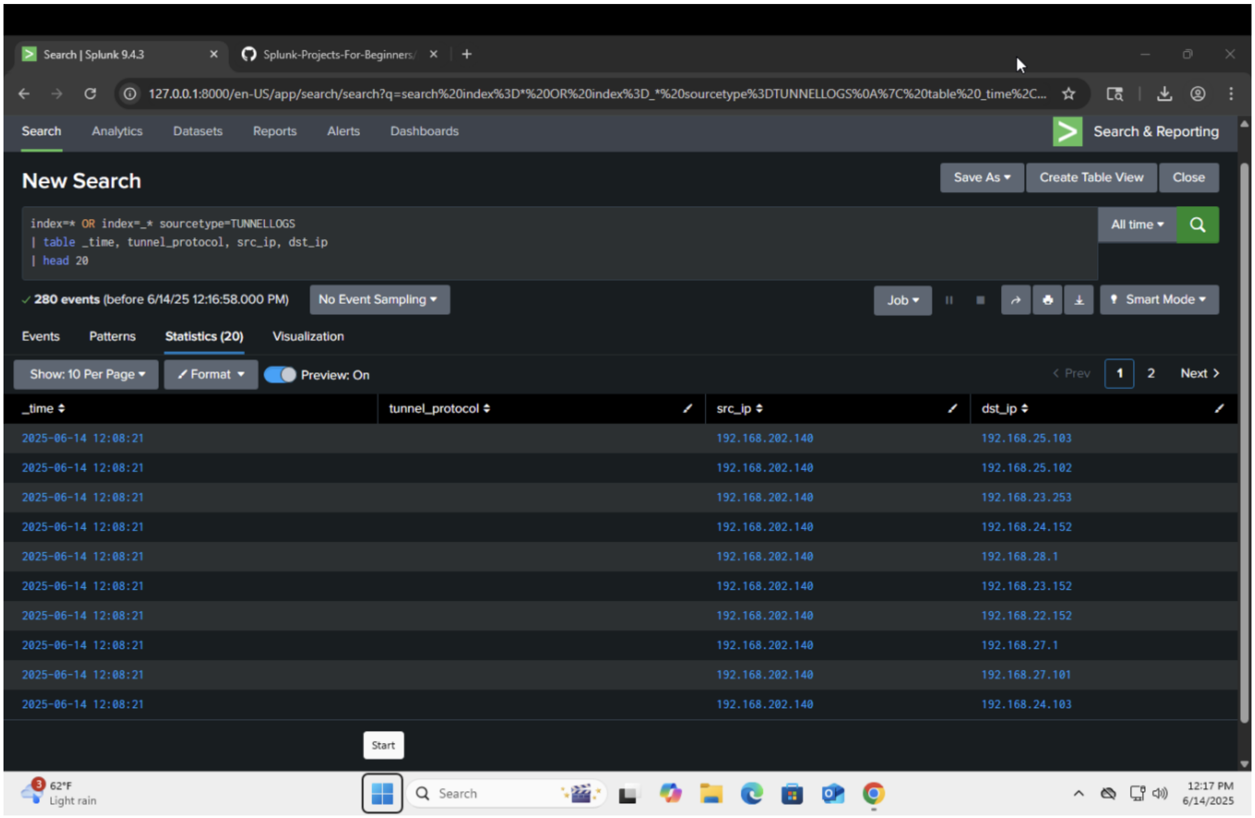
Objective:

To validate successful ingestion of Zeek tunnel logs into the Splunk platform and perform an initial exploratory data analysis by listing timestamped tunnel traffic entries. This early-stage query establishes visibility into source and destination IP pairs involved in tunneling behavior and prepares the groundwork for advanced detection use cases (e.g., detecting covert communications or policy violations).

Outcome:

- Successfully returned the **first 20 entries** from tunnel traffic logs (`sourcetype=TUNNELLOGS`) across all available indexes.
- Each entry displayed:
  - The **timestamp** of the tunnel event.
  - The **source IP** initiating the tunnel communication.
  - The **destination IP** involved in the tunnel session.
- This confirmed:
  - **Tunnel logs were successfully indexed** in Splunk.

- **Field extractions for `src_ip`, `dst_ip`, and `_time`** were correct or inferred based on Zeek log format.
- Served as a **baseline visibility check** before conducting more advanced queries like protocol classification (`TEREDO`, `AYIYA`, etc.), anomaly detection, and activity correlation.



```
index=* OR index=_* sourcetype=TUNNELLOGS
| table _time, tunnel_protocol, src_ip, dst_ip
| head 20
```

✓ **280 events** (before 6/14/25 12:16:58.000 PM)

_time	tunnel_protocol	src_ip	dst_ip
2025-06-14 12:08:21		192.168.202.140	192.168.25.103
2025-06-14 12:08:21		192.168.202.140	192.168.25.102
2025-06-14 12:08:21		192.168.202.140	192.168.23.253
2025-06-14 12:08:21		192.168.202.140	192.168.24.152
2025-06-14 12:08:21		192.168.202.140	192.168.28.1
2025-06-14 12:08:21		192.168.202.140	192.168.23.152
2025-06-14 12:08:21		192.168.202.140	192.168.22.152
2025-06-14 12:08:21		192.168.202.140	192.168.27.1
2025-06-14 12:08:21		192.168.202.140	192.168.27.101
2025-06-14 12:08:21		192.168.202.140	192.168.24.103

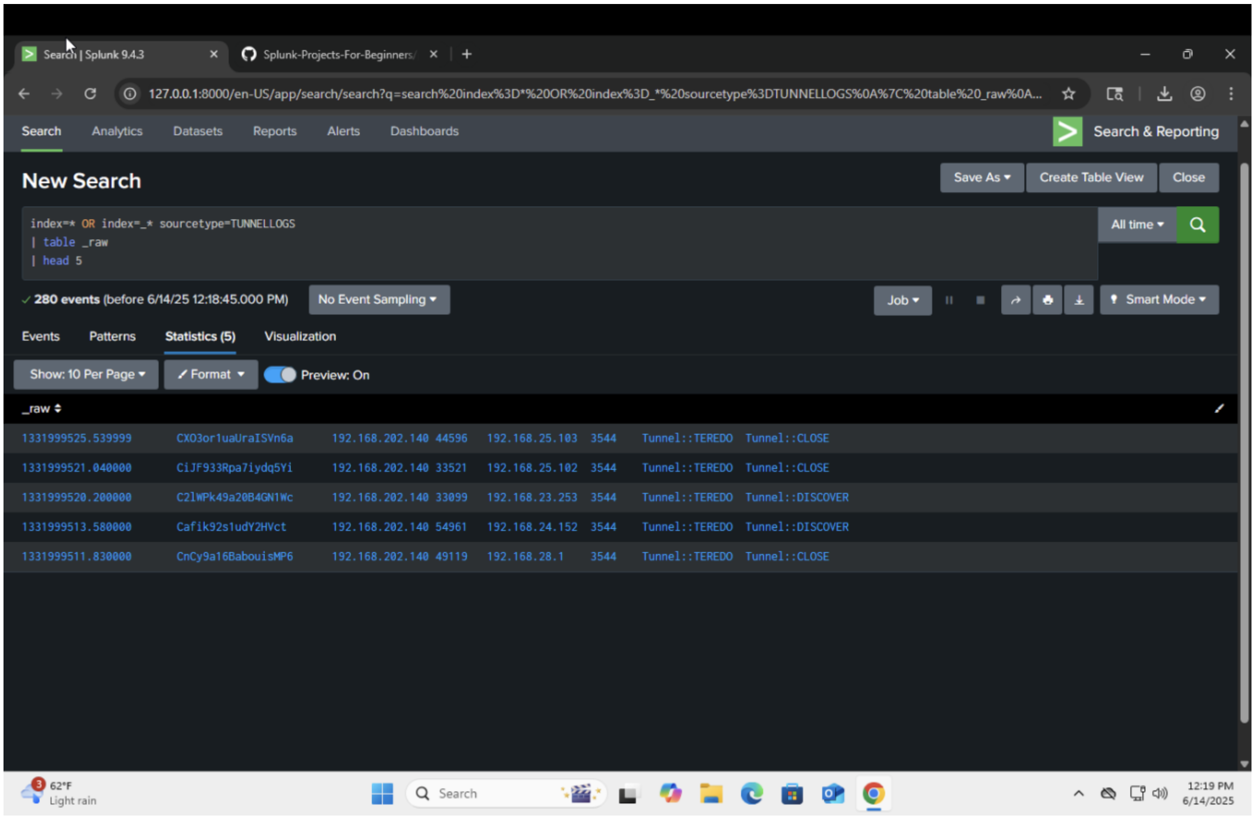
2. Inspect Raw Tunnel Log Entries

Objective:

To view raw, unparsed tunnel log events in their entirety to understand their structure and content. This step helps in designing accurate field extractions and confirms the raw data format before applying further parsing or statistical commands.

Outcome:

- Retrieved the **first 5 raw tunnel log entries** without any field extraction or transformation.
- Allowed manual inspection of the **complete log lines** exactly as ingested.
- Confirmed the logs contain rich detail such as:
    - Timestamps
    - Session IDs
    - Source and destination IPs and ports
    - Tunnel protocol markers like `Tunnel::TEREDO`
    - Tunnel states like `Tunnel::CLOSE` or `Tunnel::DISCOVER`
- This granular view informs the design of extraction patterns (e.g., regex) for downstream queries



```
index=* OR index=_* sourcetype=TUNNELLOGS
| table _raw
| head 5
```

✓ 280 events (before 6/14/25 12:18:45.000 PM) No Event Sampling

```
_raw
1331999525.539999   CXO3or1uaUraISVn6a   192.168.202.140 44596   192.168.25.103  3544   Tunnel::TEREDO  Tunnel::CLOSE
1331999521.040000   CiJF933Rpa7iydq5Yi   192.168.202.140 33521   192.168.25.102  3544   Tunnel::TEREDO  Tunnel::CLOSE
1331999520.200000   C2lWPk49a20B4GN1Wc   192.168.202.140 33099   192.168.23.253  3544   Tunnel::TEREDO  Tunnel::DISCOVER
1331999513.580000   Cafik92s1udY2HVct    192.168.202.140 54961   192.168.24.152  3544   Tunnel::TEREDO  Tunnel::DISCOVER
1331999511.830000   CnCy9a16BabouisMP6   192.168.202.140 49119   192.168.28.1    3544   Tunnel::TEREDO  Tunnel::CLOSE
```

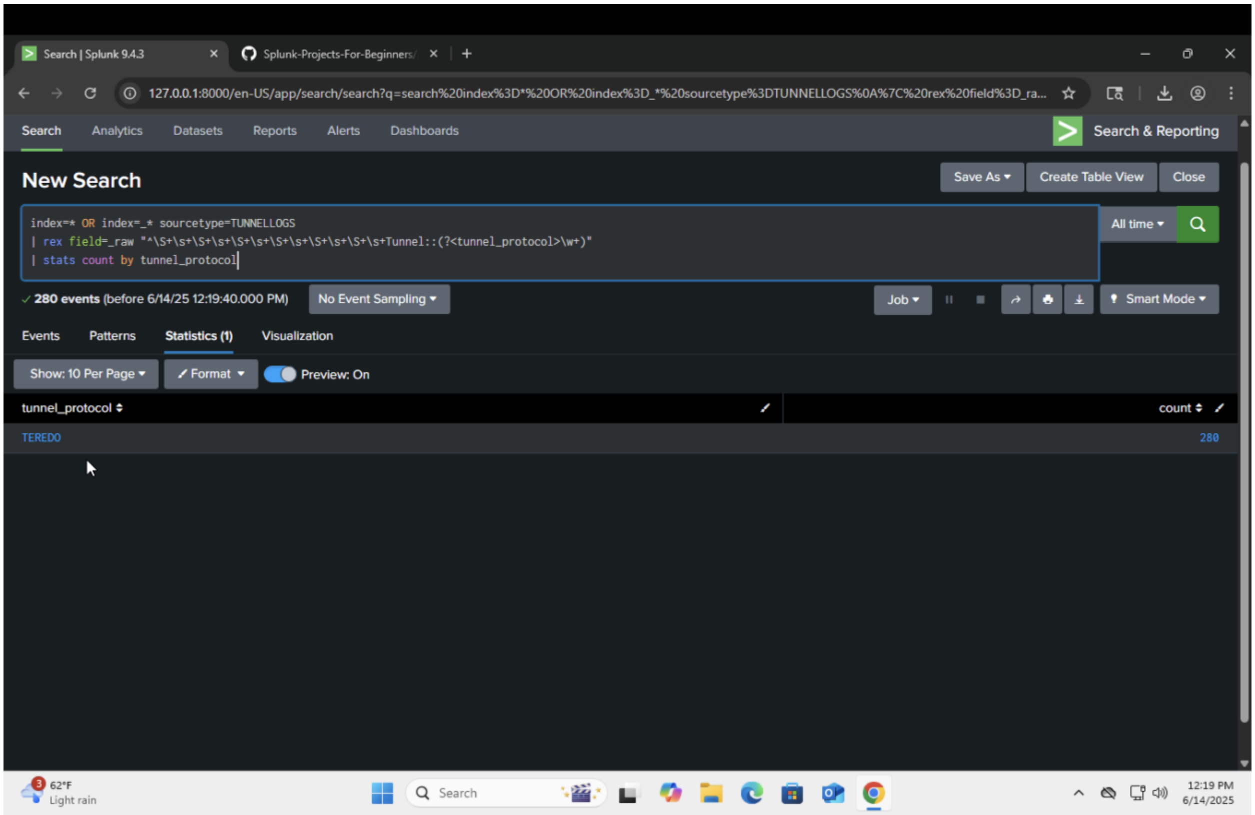
3. Count Tunnel Protocol Occurrences

Objective:

Extract the tunnel protocol from raw tunnel logs and count how many times each protocol appears. This helps identify the dominant tunneling protocols in the network traffic.

Outcome:

- Successfully extracted the **`tunnel_protocol`** field from the raw log lines.
- Counted the occurrences of each tunneling protocol.
- Example result: `TEREDO` appeared 280 times, indicating it's the most common tunnel protocol in the data set.
- This insight lets you focus on the protocols actually in use and prioritize analysis or anomaly detection accordingly.



```
index=* OR index=_* sourcetype=TUNNELLOGS
| rex field=_raw "^\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+Tunnel::(?<tunnel_protocol>\w+)"
| stats count by tunnel_protocol
```

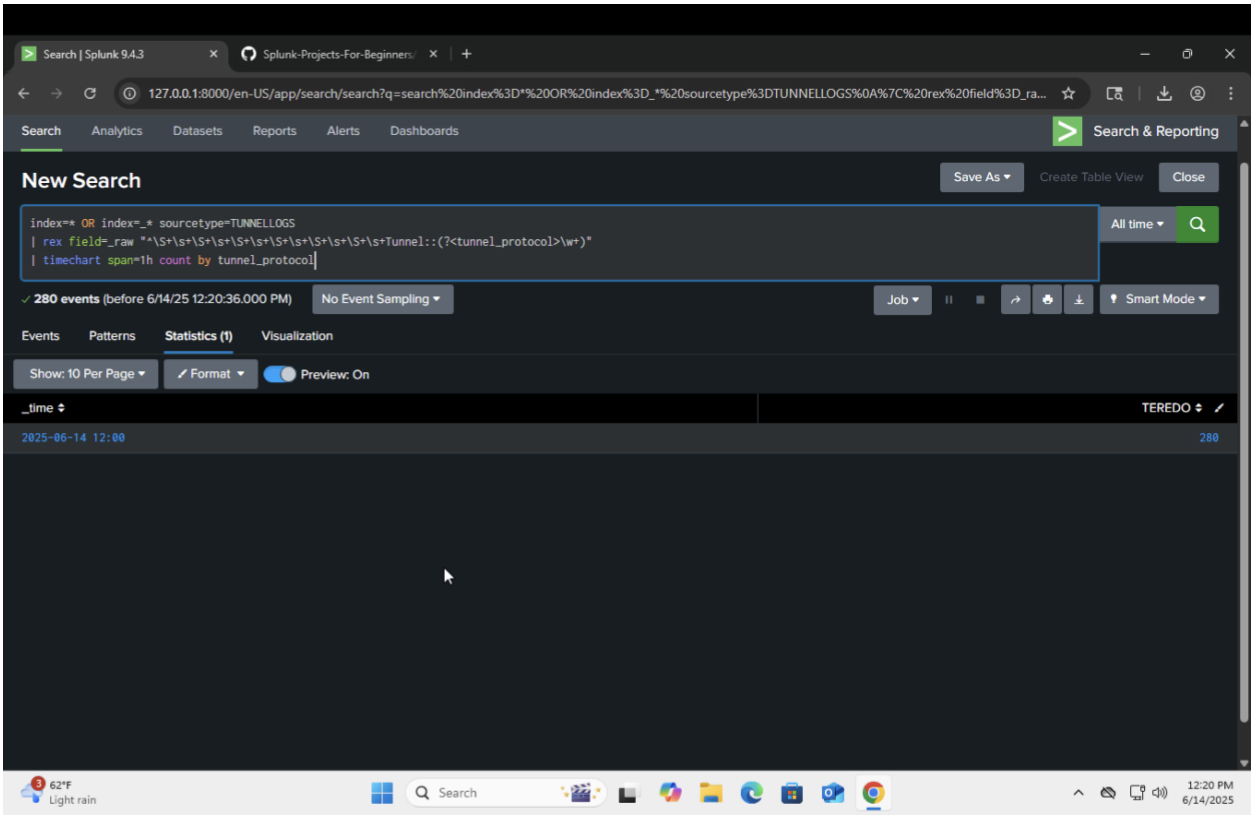
✓ 280 events (before 6/14/25 12:19:40.000 PM)

tunnel_protocol	count
TEREDO	280

4. Time-Based Tunnel Protocol Activity Chart

Objective:
To visualize how tunnel protocol usage changes over time, aggregated hourly.

Outcome:
Created an hourly time chart of tunnel protocol activity, which helps identify spikes or unusual activity periods. This visualization is crucial for spotting when specific tunneling protocols become more or less active.



New Search

```
index=* OR index=_* sourcetype=TUNNELLOGS
| rex field=_raw "^\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+Tunnel::(?<tunnel_protocol>\w+)"
| timechart span=1h count by tunnel_protocol
```

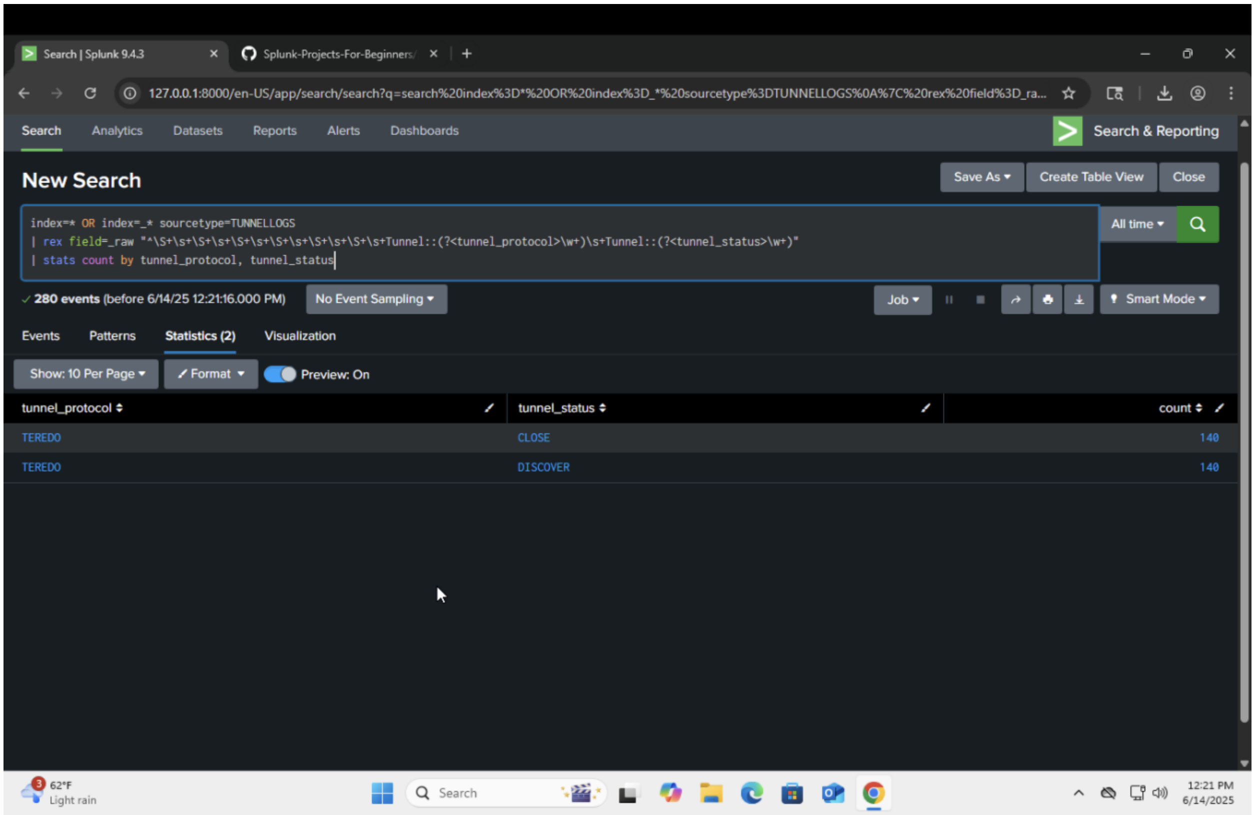
✓ 280 events (before 6/14/25 12:20:36.000 PM) No Event Sampling

_time	TEREDO
2025-06-14 12:00	280

5. Extract Tunnel Protocol and Status Counts

Objective:
To extract the tunnel protocol and tunnel status fields from raw logs and count how many times each protocol-status combination appears.

Outcome:
Successfully extracted `tunnel_protocol` (e.g., TEREDO) and `tunnel_status` (e.g., CLOSE, DISCOVER) and generated a count summary. This helps identify the most common protocols and their states in your logs.



```
index=* OR index=_* sourcetype=TUNNELLOGS
| rex field=_raw "^\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+Tunnel::(?<tunnel_protocol>\w+)\s+Tunnel::(?<tunnel_status>\w+)"
| stats count by tunnel_protocol, tunnel_status
```

✓ 280 events (before 6/14/25 12:21:16.000 PM)

tunnel_protocol	tunnel_status	count
TEREDO	CLOSE	140
TEREDO	DISCOVER	140

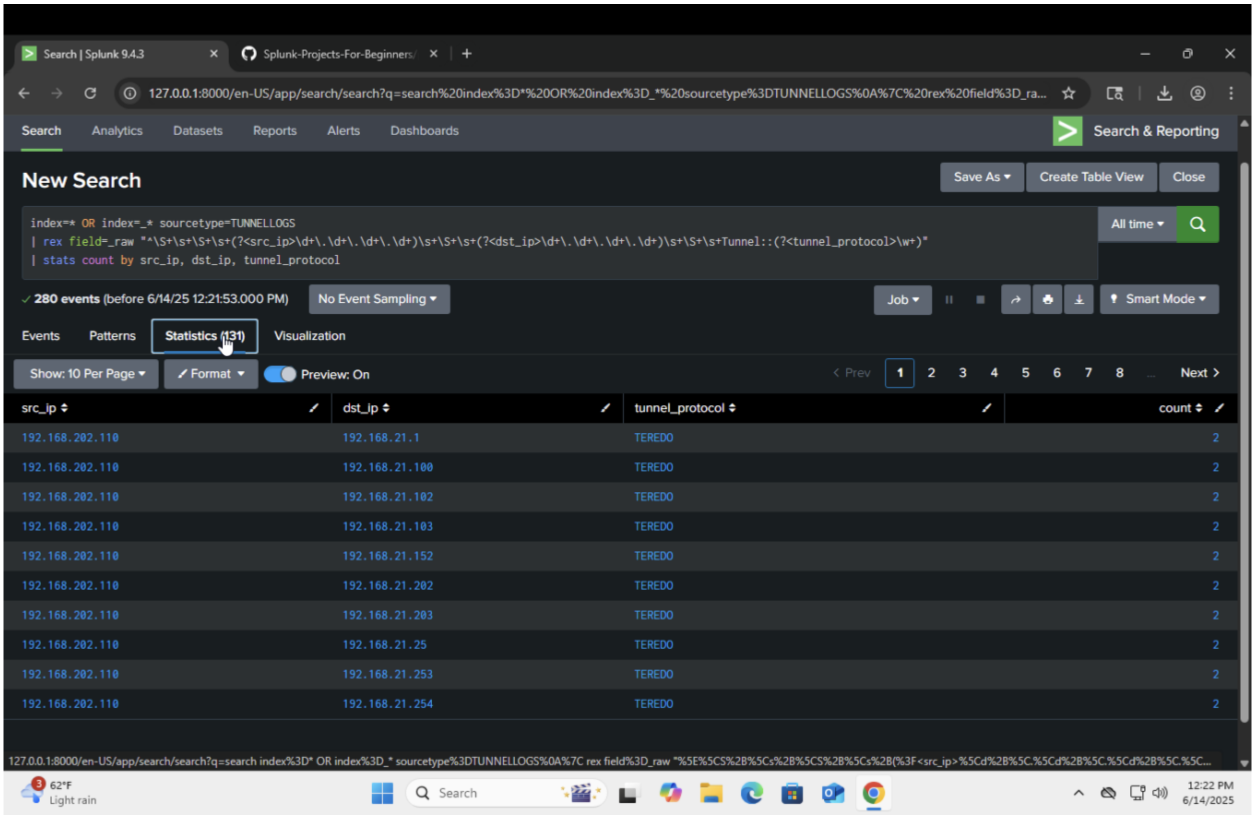
6. Analyze Tunnel Traffic by Protocol and IP Addresses

Objective:

Extract tunnel protocol and status, then count the number of log events grouped by tunnel protocol, source IP, and destination IP. This helps map which hosts are using which tunneling protocols and the volume of their traffic, essential for spotting unusual or suspicious connections.

Outcome:

- Successfully extracted **`tunnel_protocol`** and **`tunnel_status`** from raw logs.
- Aggregated counts of tunneling events by protocol and by source and destination IP addresses.
- Provided a detailed map of which internal hosts (source IPs) are communicating with which external/internal hosts (destination IPs) using specific tunneling protocols.
- This data enables detection of suspicious or high-volume tunnels potentially indicating unauthorized data exfiltration or lateral movement inside the network.



Search | Splunk 9.4.3
Splunk-Projects-For-Beginners/
127.0.0.1:8000/en-US/app/search/search?q=search%20index%3D*%20OR%20index%3D_*%20sourcetype%3DTUNNELLOGS%0A%7C%20rex%20field%3D_ra...
Search
Analytics
Datasets
Reports
Alerts
Dashboards
Search & Reporting
New Search
Save As
Create Table View
Close
All time
No Event Sampling
Job
Smart Mode
Events
Patterns
Statistics (131)
Visualization
Show: 10 Per Page
Format
Preview: On
Prev 1 2 3 4 5 6 7 8 ... Next
127.0.0.1:8000/en-US/app/search/search?q=search index%3D* OR index%3D_* sourcetype%3DTUNNELLOGS%0A%7C rex field%3D_raw "%5E%5CS%2B%5Cs%2B%5CS%2B%5Cs%2B(%3F<src_ip>%5Cd%2B%5C.%5Cd%2B%5C.%5Cd%2B%5C.%5C...
62°F
Light rain
Search
12:22 PM
6/14/2025

```
index=* OR index=_* sourcetype=TUNNELLOGS
| rex field=_raw "^\S+\s+\S+\s+(?<src_ip>\d+\.\d+\.\d+\.\d+)\s+\S+\s+(?<dst_ip>\d+\.\d+\.\d+\.\d+)\s+\S+\s+Tunnel::(?<tunnel_protocol>\w+)"
| stats count by src_ip, dst_ip, tunnel_protocol
```

✓ 280 events (before 6/14/25 12:21:53.000 PM)

src_ip	dst_ip	tunnel_protocol	count
192.168.202.110	192.168.21.1	TEREDO	2
192.168.202.110	192.168.21.100	TEREDO	2
192.168.202.110	192.168.21.102	TEREDO	2
192.168.202.110	192.168.21.103	TEREDO	2
192.168.202.110	192.168.21.152	TEREDO	2
192.168.202.110	192.168.21.202	TEREDO	2
192.168.202.110	192.168.21.203	TEREDO	2
192.168.202.110	192.168.21.25	TEREDO	2
192.168.202.110	192.168.21.253	TEREDO	2
192.168.202.110	192.168.21.254	TEREDO	2

7. Extract Source IP and Tunnel Protocol with Count Sorting

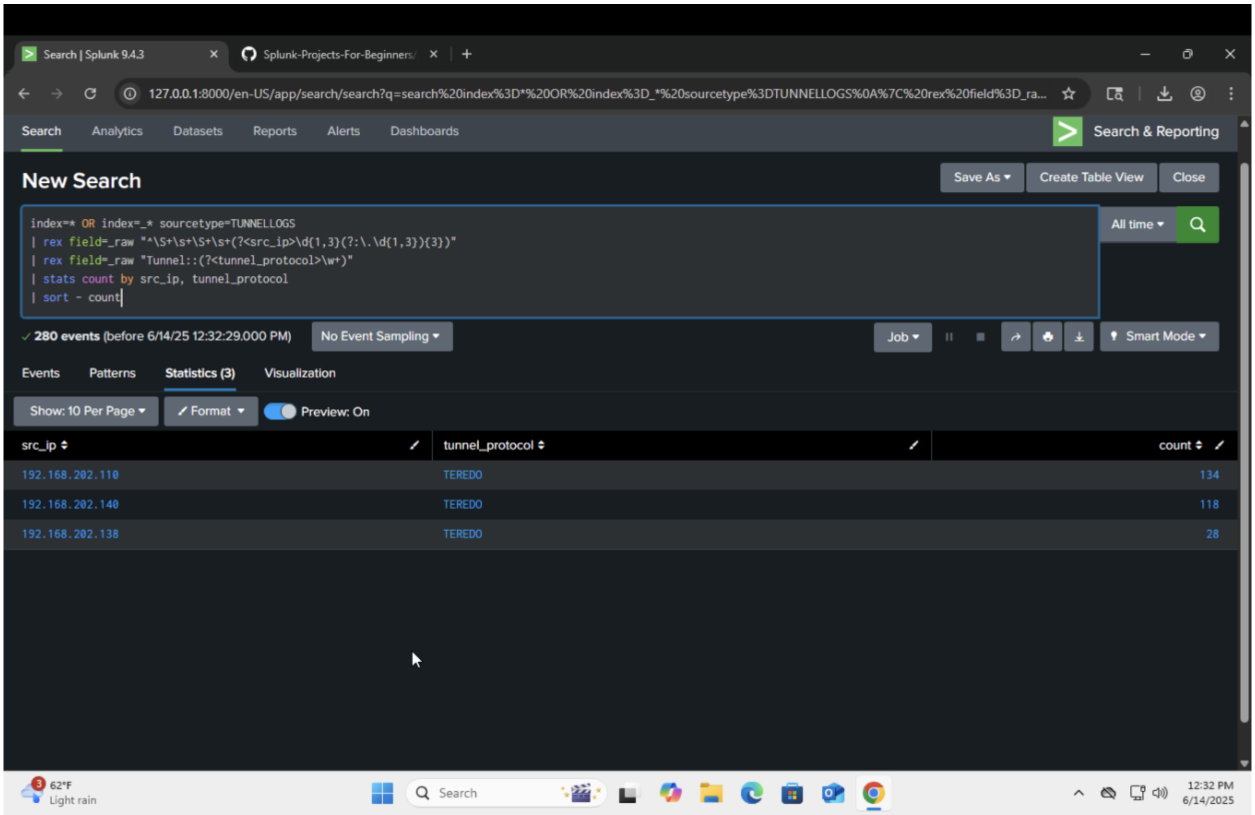
Objective:

- Extract the **source IP address** and **tunnel protocol** from the raw tunnel logs.
- Count the number of events per tunnel protocol and source IP.
- Sort the results in descending order by count to highlight the most active sources and protocols.

Outcome:

- Successfully extracted the `src_ip` and `tunnel_protocol` fields from raw log entries.
- Produced a count of tunnel activity grouped by source IP and protocol.
- Sorted the data so the most frequent tunnel protocol and source IP combinations appear at the top.

- This highlights the heavy hitters — sources generating the most tunnel traffic per protocol, key for spotting anomalies or compromised hosts.



```
index=* OR index=_* sourcetype=TUNNELLOGS
| rex field=_raw "^\S+\s+\S+\s+(?<src_ip>\d{1,3}(?:\.\d{1,3}){3})"
| rex field=_raw "Tunnel::(?<tunnel_protocol>\w+)"
| stats count by src_ip, tunnel_protocol
| sort - count
```

✓ 280 events (before 6/14/25 12:32:29.000 PM) No Event Sampling

src_ip	tunnel_protocol	count
192.168.202.110	TEREDO	134
192.168.202.140	TEREDO	118
192.168.202.138	TEREDO	28

Conclusion

This project confirmed the robust ingestion of Zeek tunnel logs into Splunk and established foundational visibility through precise field extractions. By quantifying tunnel protocols and visualizing their activity over time, we identified TEREDO as the dominant tunneling protocol and detected traffic patterns indicative of normal and potentially suspicious activity. Mapping traffic flows by IP addresses provided crucial insights into host communication behaviors, equipping analysts with actionable data to detect covert tunnels, policy violations, or lateral movement within the network. These findings lay the groundwork for proactive threat hunting and enhanced network security monitoring using Splunk.