LOG ANALYSIS USING SPLUNK

By

T Sharan Sebastian

CONTENTS

ABOUT

A **Log Analysis and Security Operations Center (SOC) project** involves the centralized collection, monitoring, and analysis of logs generated by various sources such as operating systems, network devices, firewalls, servers, and security tools to identify potential security threats and operational issues. In this project, logs are ingested into a SIEM platform where they are normalized, correlated, and analyzed in real time to detect suspicious activities such as brute-force login attempts, malware behavior, unauthorized access, or web-based attacks. The SOC environment simulates real-world security operations by enabling continuous monitoring through dashboards, automated alert generation, and structured investigation workflows. Security alerts are reviewed by analysts who perform triage to distinguish false positives from real incidents, investigate indicators of compromise using log data and threat intelligence, and initiate response actions such as blocking malicious IP addresses or isolating affected systems. The project also emphasizes incident documentation, reporting, and improvement of detection rules, demonstrating how log analysis plays a critical role in proactive threat detection, incident response, and maintaining an organization's overall cybersecurity posture.

The goal of this project is to analyze SSH authentication logs to detect:

- **Successful logins** (who connected, from where)
- **Failed login attempts** (possible brute-force or password spraying)
- **Multiple failed authentication attempts** (indicators of brute-force)
- **Connections without authentication** (potential scanning or incomplete sessions)

SIEM (Security Information and Event Management)

A **SIEM (Security Information and Event Management)** system is a core component of a SOC that collects, normalizes, and correlates log data from multiple sources such as servers, endpoints, firewalls, applications, and network devices to provide centralized security monitoring. SIEM tools analyze large volumes of log data in real time to detect anomalies, security threats, and policy violations by applying correlation rules, behavioral analysis, and threat intelligence. Commonly used SIEM tools include **Splunk**, **ELK Stack (Elasticsearch, Logstash, Kibana)**, **Wazuh**, **IBM QRadar**, and **Microsoft Sentinel**, each offering features like real-time alerting, dashboards, and incident investigation capabilities. Supporting tools such as **Snort or Suricata** (IDS/IPS), **Zeek** (network traffic analysis), **Wireshark** (packet analysis), and **VirusTotal** (threat intelligence) work alongside SIEM platforms to enhance detection and analysis. Together, SIEM and these security tools enable SOC teams to monitor environments effectively, respond to incidents quickly, and improve an organization's overall security visibility and resilience.

splunk is a powerful SIEM tool that collects and indexes large volumes of log data from systems, applications, and network devices, allowing analysts to search, correlate, and visualize events through dashboards and alerts. It is widely used for real-time monitoring, threat detection, and compliance reporting.

Wazuh is an open-source security platform that provides log analysis, host-based intrusion detection (HIDS), file integrity monitoring, vulnerability detection, and endpoint security by collecting detailed data from agents installed on endpoints.

IBM QRadar is an enterprise-grade SIEM that focuses on advanced log correlation and offense management, while **Microsoft Sentinel** is a cloud-based SIEM that integrates closely with Azure services and uses analytics and automation for threat detection and response. Together, these tools help SOC teams achieve centralized visibility, efficient threat detection, and effective incident response.

LOG ANALYSIS

Log analysis is the process of collecting, reviewing, and analyzing log data generated by systems, applications, servers, network devices, and security tools to identify security threats, system issues, and unusual behavior. Logs record important events such as user logins, file access, errors, network connections, and application activities. By analyzing these logs, security analysts can detect attacks like brute-force login attempts, malware infections, unauthorized access, and data exfiltration. In a SOC environment, log analysis is performed using SIEM tools that centralize logs, correlate events from multiple sources, and generate alerts for suspicious patterns. Effective log analysis helps organizations improve threat detection, investigate incidents, ensure compliance, and maintain overall system security and stability.

Types of logs

Understanding the types of logs you'll encounter can help you analyze logs more effectively. Each type serves a unique purpose and offers different insights.

Here are some types of logs you may encounter:

Access logs

Access logs record every request made to a server, including details like IP addresses, timestamps, requested resources, and response codes. These logs are vital for understanding user behavior, tracking traffic patterns, and identifying potential security threats.

For instance, if you notice a sudden spike in requests from a single IP address, it could indicate a potential DDoS attack. Analyzing access logs helps you take proactive measures to safeguard your systems.

Error logs

Error logs capture incidents where something went wrong within a system or application. This could include failed database connections, missing files, or crashed applications. These logs are invaluable for troubleshooting and ensuring the smooth operation of your services.

For such logs, you can analyze these errors and resolve any issues before they escalate, minimizing downtime and enhancing user experience.

Event logs

Event logs provide a comprehensive record of significant system events, such as user logins, system startup, and configuration changes. They are essential for maintaining system integrity and compliance.

For example, event logs can help you trace unauthorized access attempts or track changes made to critical system settings, ensuring you maintain control over your environment.

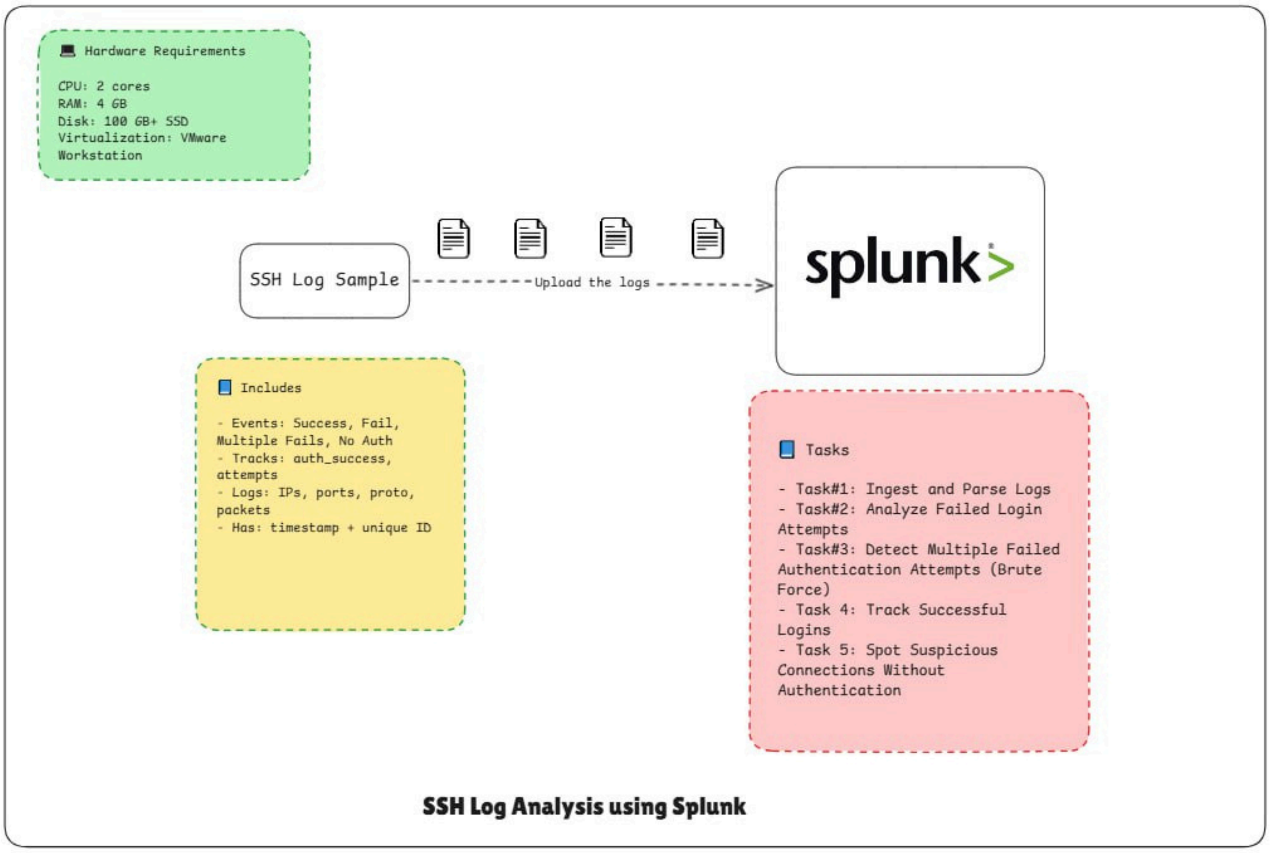
SPLUNK

Splunk is a powerful data analytics and security platform used to collect, index, search, and analyze machine-generated data such as logs from servers, applications, network devices, and cloud services. In cybersecurity, Splunk is commonly used as a SIEM solution to monitor security events, detect threats like failed login attempts, brute-force attacks, suspicious connections, and successful logins after failures, and support incident response. It provides real-time visibility, advanced correlation, dashboards, alerts, and integrations with security tools, helping organizations improve threat detection, compliance, and operational intelligence across on-premises and cloud environments.

SSH LOG ANALYSIS

SSH (Secure Shell) is a secure network protocol used to remotely access and manage systems over an encrypted connection, commonly used by administrators to control Linux and Unix servers. **SSH logs** record all SSH-related activities on a system, such as successful and failed login attempts, authentication methods used, user accounts involved, source IP addresses, and session start or end times. These logs are typically stored in files like /var/log/auth.log or /var/log/secure on Linux systems.

In **log analysis and SOC operations**, SSH logs are extremely important for detecting security threats such as brute-force attacks, unauthorized access attempts, credential abuse, or compromised accounts. Analysts monitor SSH logs for repeated failed login attempts from the same IP, logins from unusual locations, access outside normal working hours, or use of root accounts. By analyzing SSH logs through SIEM tools, security teams can generate alerts, investigate suspicious behavior, block malicious IP addresses, and strengthen access controls, helping to protect critical systems from remote attacks.

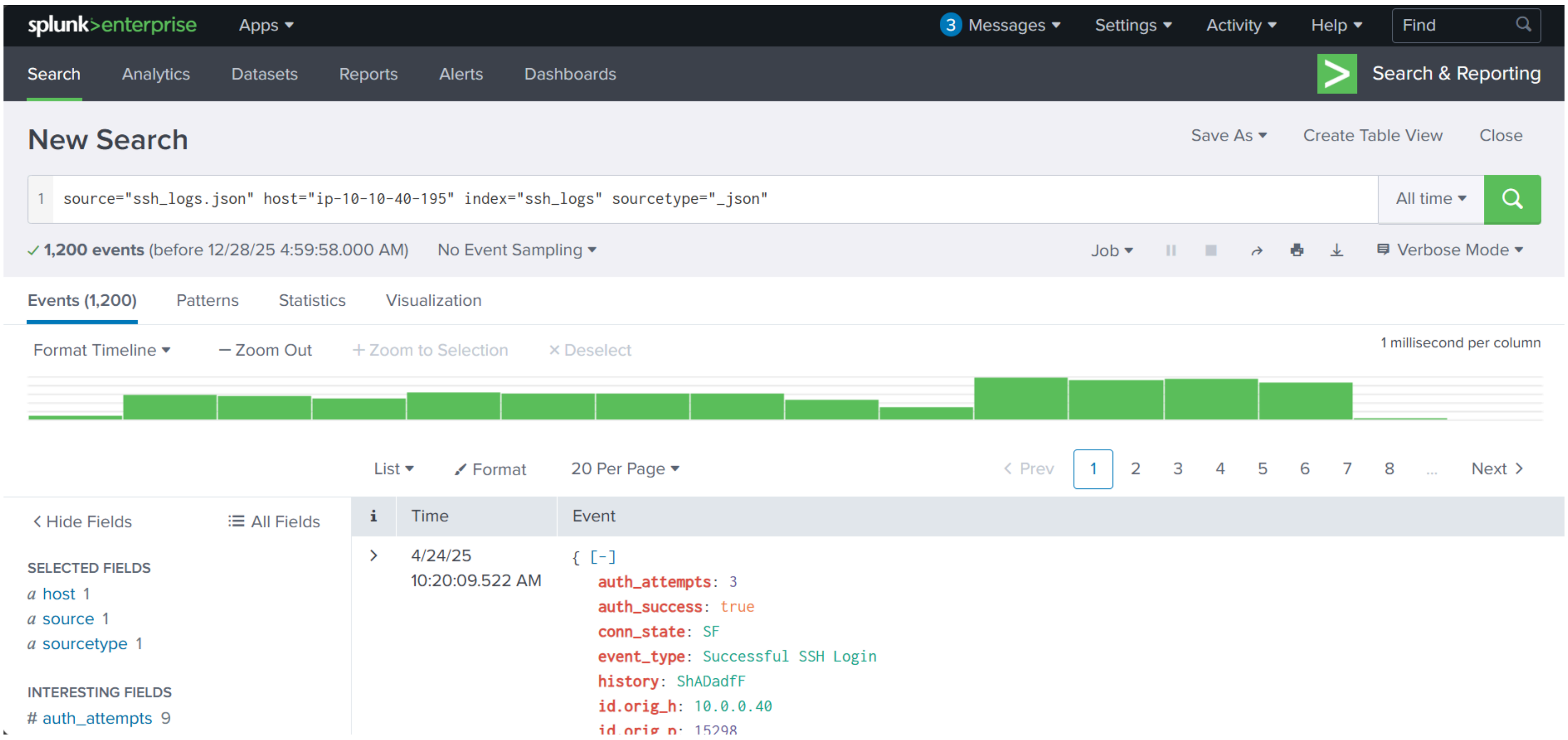

Hardware Requirements
CPU: 2 cores
RAM: 4 GB
Disk: 100 GB+ SSD
Virtualization: VMware Workstation
SSH Log Sample
Upload the logs
splunk>
Includes
- Events: Success, Fail, Multiple Fails, No Auth
- Tracks: auth_success, attempts
- Logs: IPs, ports, proto, packets
- Has: timestamp + unique ID
Tasks
- Task#1: Ingest and Parse Logs
- Task#2: Analyze Failed Login Attempts
- Task#3: Detect Multiple Failed Authentication Attempts (Brute Force)
- Task 4: Track Successful Logins
- Task 5: Spot Suspicious Connections Without Authentication
SSH Log Analysis using Splunk

PROCEDURE

STEP - 1: INGEST AND PARSE THE LOG

Ingesting the logs in SEIM tool. Here I used the tool called Splunk. These fields are going to help to analyse the logs.

- **event_type** (Successful SSH Login, Failed SSH Login, Multiple Failed Authentication Attempts, Connection Without Authentication)
- **auth_success** (true/false/null)
- **auth_attempts**
- **id.orig_h** (source IP)
- **id.resp_h** (destination host)



```
source="ssh_logs.json" host="ip-10-10-40-195" index="ssh_logs" sourcetype="_json"
```

✓ 1,200 events (before 12/28/25 4:59:58.000 AM) No Event Sampling

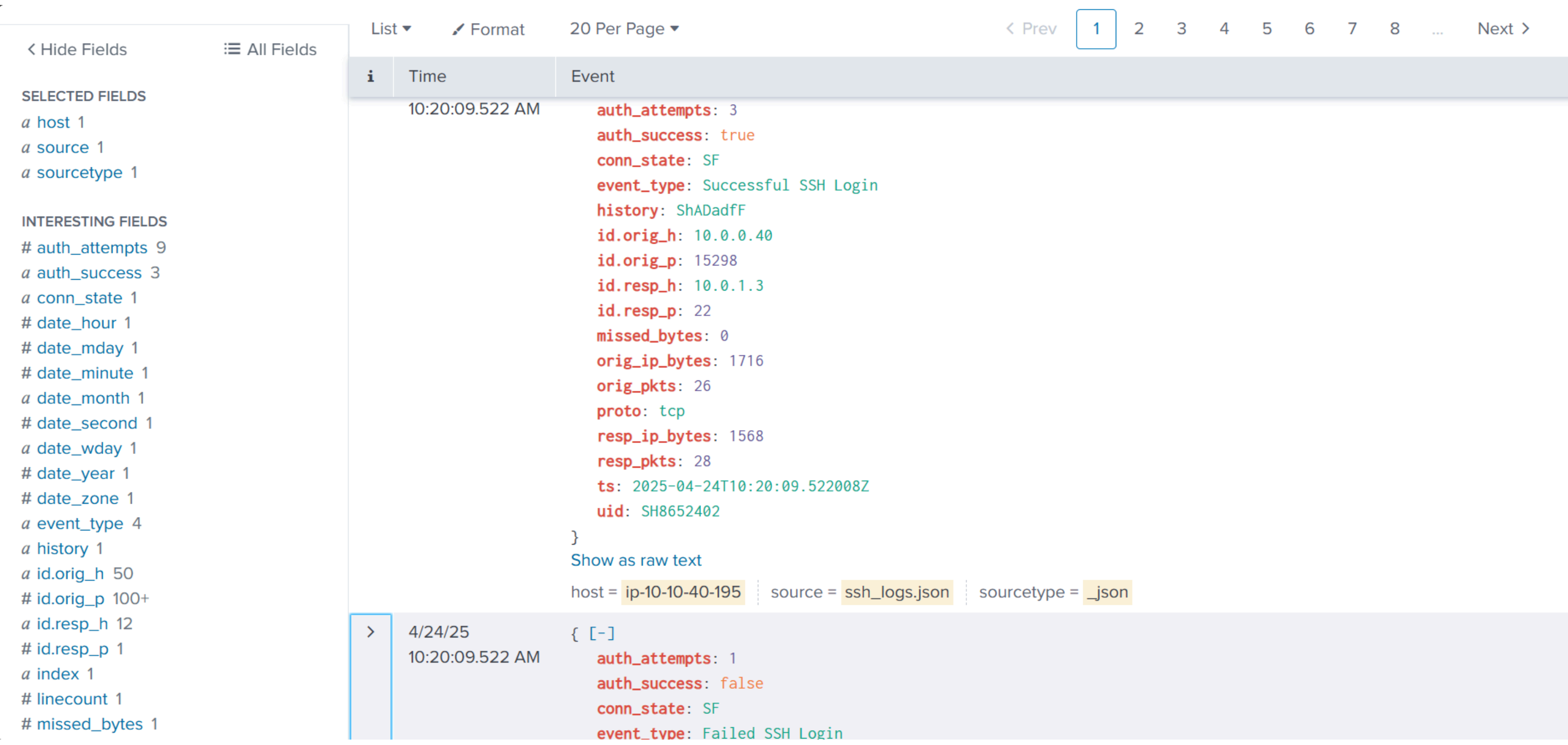
SELECTED FIELDS
a host 1
a source 1
a sourcetype 1

INTERESTING FIELDS
# auth_attempts 9

i Time Event

> 4/24/25 10:20:09.522 AM

```
{ [-]
   auth_attempts: 3
   auth_success: true
   conn_state: SF
   event_type: Successful SSH Login
   history: ShADadfF
   id.orig_h: 10.0.0.40
   id.orig_p: 15298
```



SELECTED FIELDS
a host 1
a source 1
a sourcetype 1

INTERESTING FIELDS
# auth_attempts 9
a auth_success 3
a conn_state 1
# date_hour 1
# date_mday 1
# date_minute 1
a date_month 1
# date_second 1
a date_wday 1
# date_year 1
# date_zone 1
a event_type 4
a history 1
a id.orig_h 50
# id.orig_p 100+
a id.resp_h 12
# id.resp_p 1
a index 1
# linecount 1
# missed_bytes 1

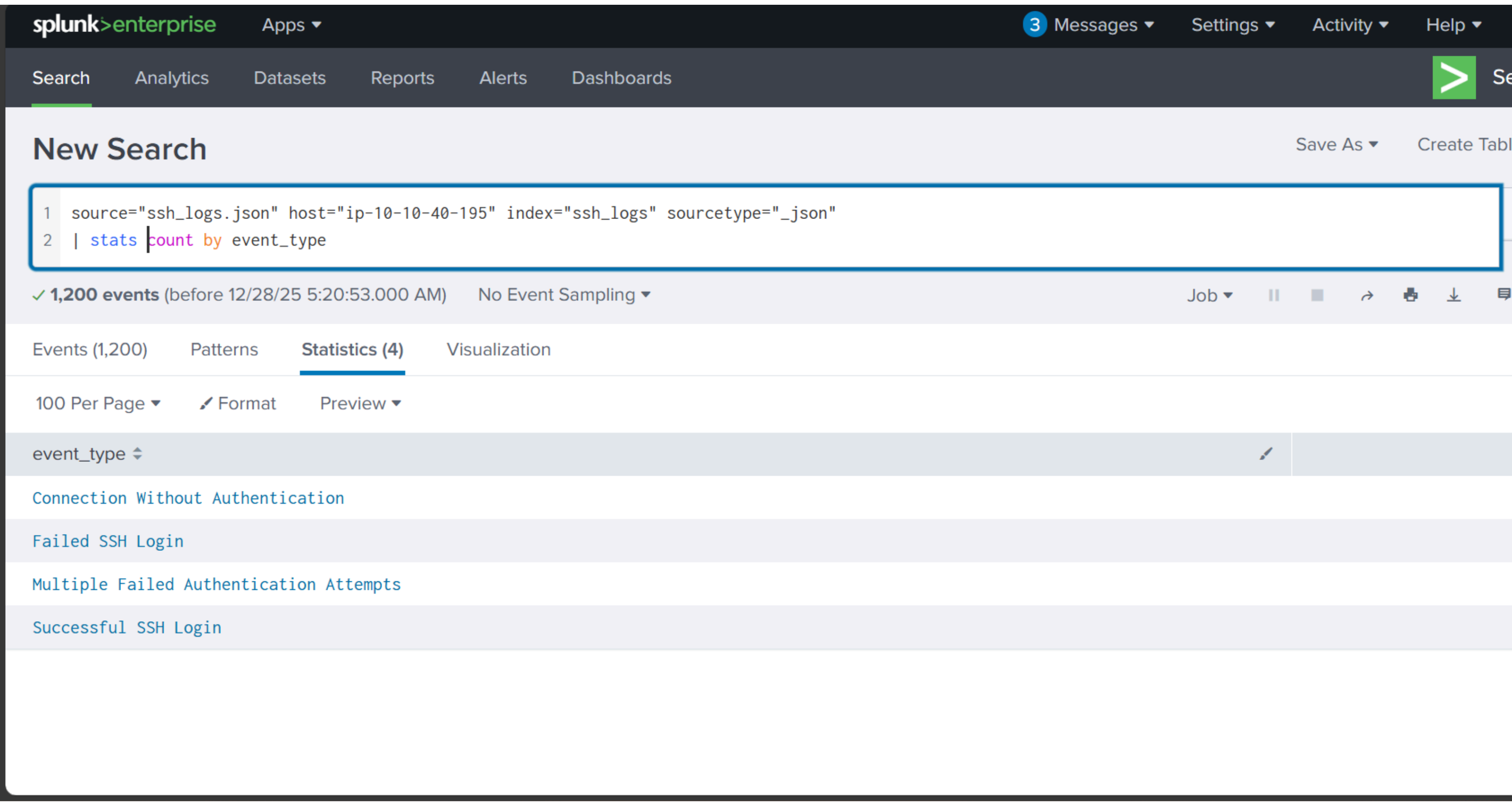
i	Time	Event

```
10:20:09.522 AM
   auth_attempts: 3
   auth_success: true
   conn_state: SF
   event_type: Successful SSH Login
   history: ShADadfF
   id.orig_h: 10.0.0.40
   id.orig_p: 15298
   id.resp_h: 10.0.1.3
   id.resp_p: 22
   missed_bytes: 0
   orig_ip_bytes: 1716
   orig_pkts: 26
   proto: tcp
   resp_ip_bytes: 1568
   resp_pkts: 28
   ts: 2025-04-24T10:20:09.522008Z
   uid: SH8652402
}
Show as raw text
host = ip-10-10-40-195   source = ssh_logs.json   sourcetype = _json

>  4/24/25             { [-]
   10:20:09.522 AM        auth_attempts: 1
                          auth_success: false
                          conn_state: SF
                          event_type: Failed SSH Login
```

After ingesting the logs, verifying that fields are extracted correctly, by running a validation check by searching this query.

```
stats count by event_type
```



```
source="ssh_logs.json" host="ip-10-10-40-195" index="ssh_logs" sourcetype="_json"
| stats count by event_type
```

✓ 1,200 events (before 12/28/25 5:20:53.000 AM)

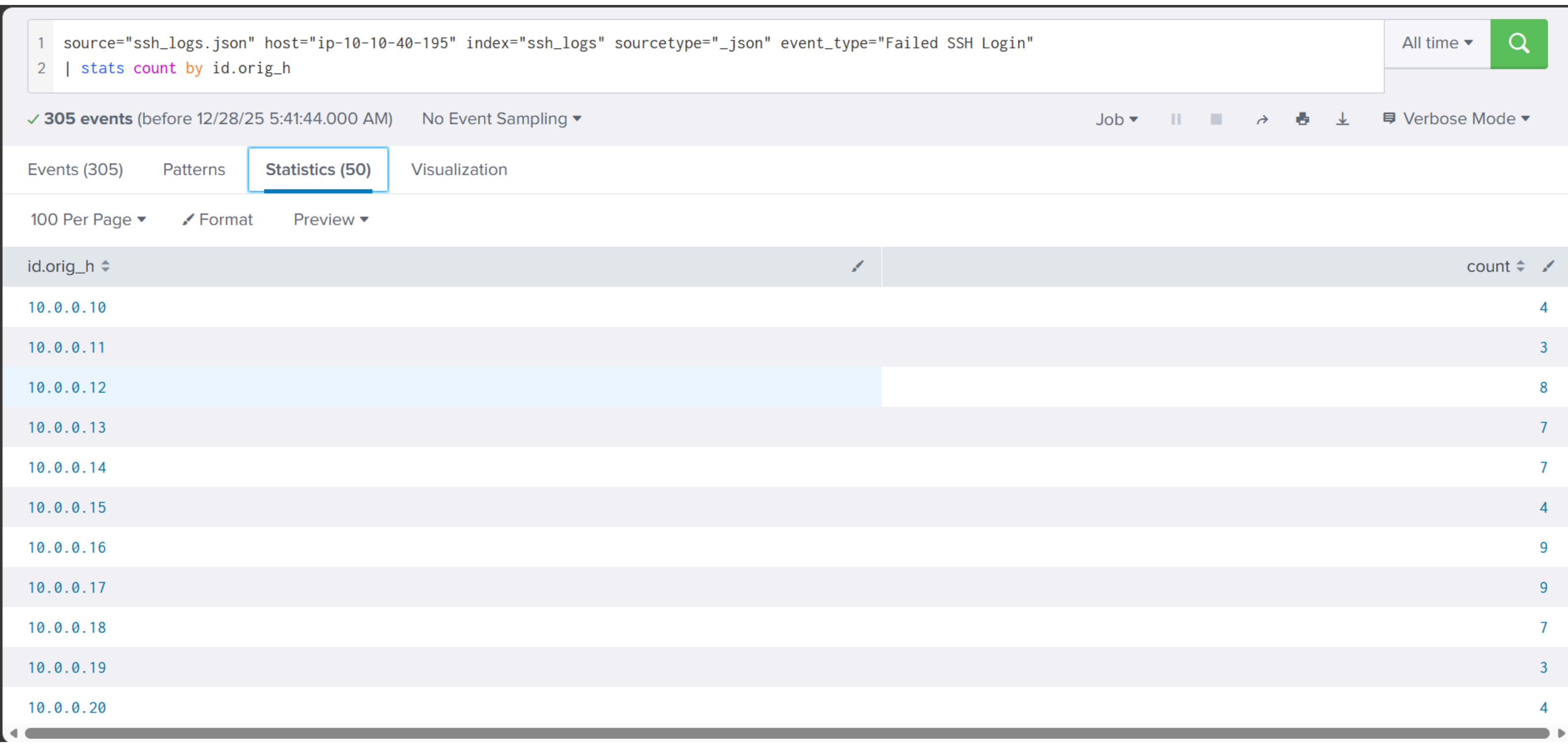
event_type
Connection Without Authentication
Failed SSH Login
Multiple Failed Authentication Attempts
Successful SSH Login

STEP - 2: ANALYSING FAILED ATTEMPTS

query:

event_type=failed ssh login” | stats count by id.orig_h

Running this query in the search bar highlights the source IP’s of failed attempts.



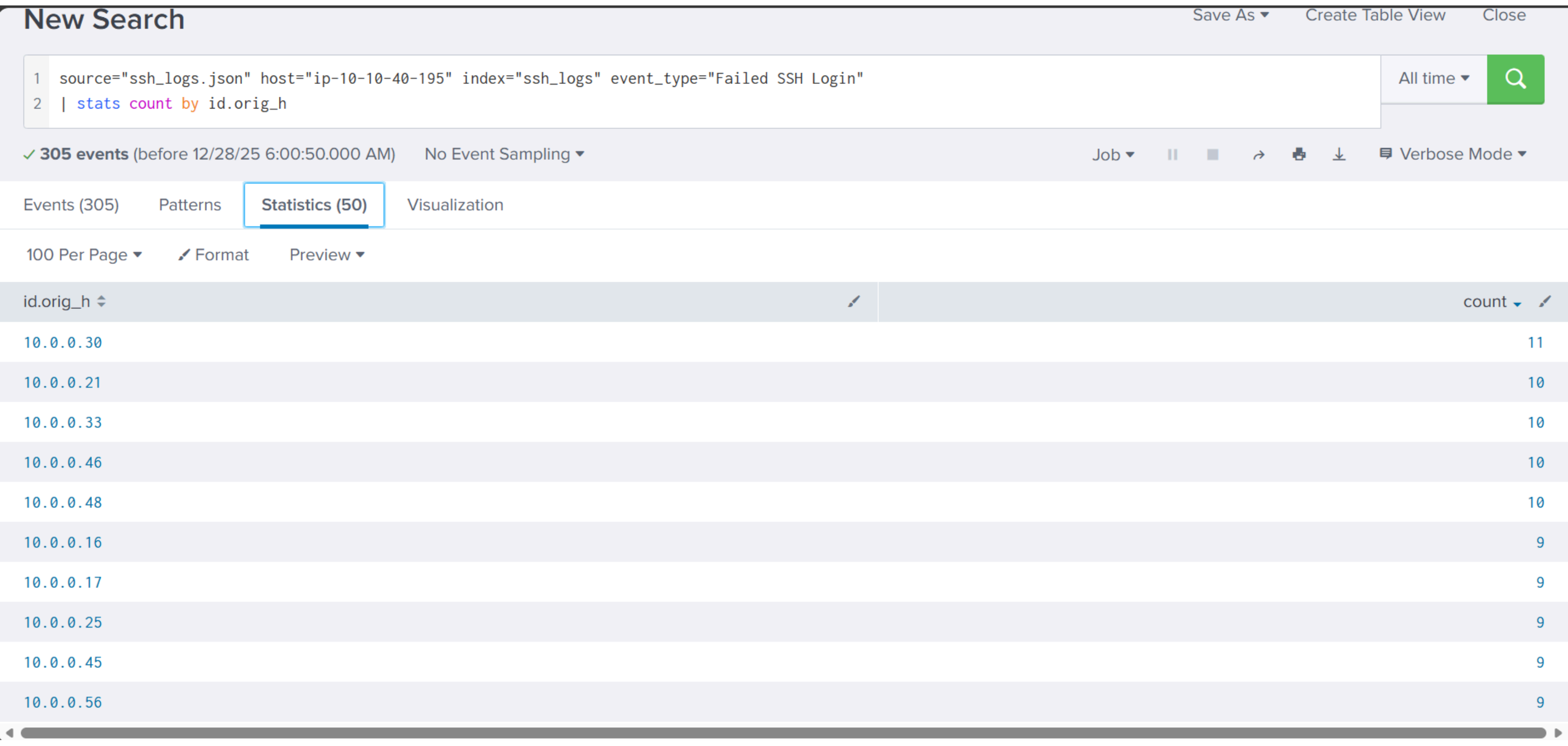
```
source="ssh_logs.json" host="ip-10-10-40-195" index="ssh_logs" sourcetype="_json" event_type="Failed SSH Login"
| stats count by id.orig_h
```

✓ **305 events** (before 12/28/25 5:41:44.000 AM) No Event Sampling

Events (305) Patterns Statistics (50) Visualization

100 Per Page Format Preview

id.orig_h	count
10.0.0.10	4
10.0.0.11	3
10.0.0.12	8
10.0.0.13	7
10.0.0.14	7
10.0.0.15	4
10.0.0.16	9
10.0.0.17	9
10.0.0.18	7
10.0.0.19	3
10.0.0.20	4



New Search

```
source="ssh_logs.json" host="ip-10-10-40-195" index="ssh_logs" event_type="Failed SSH Login"
| stats count by id.orig_h
```

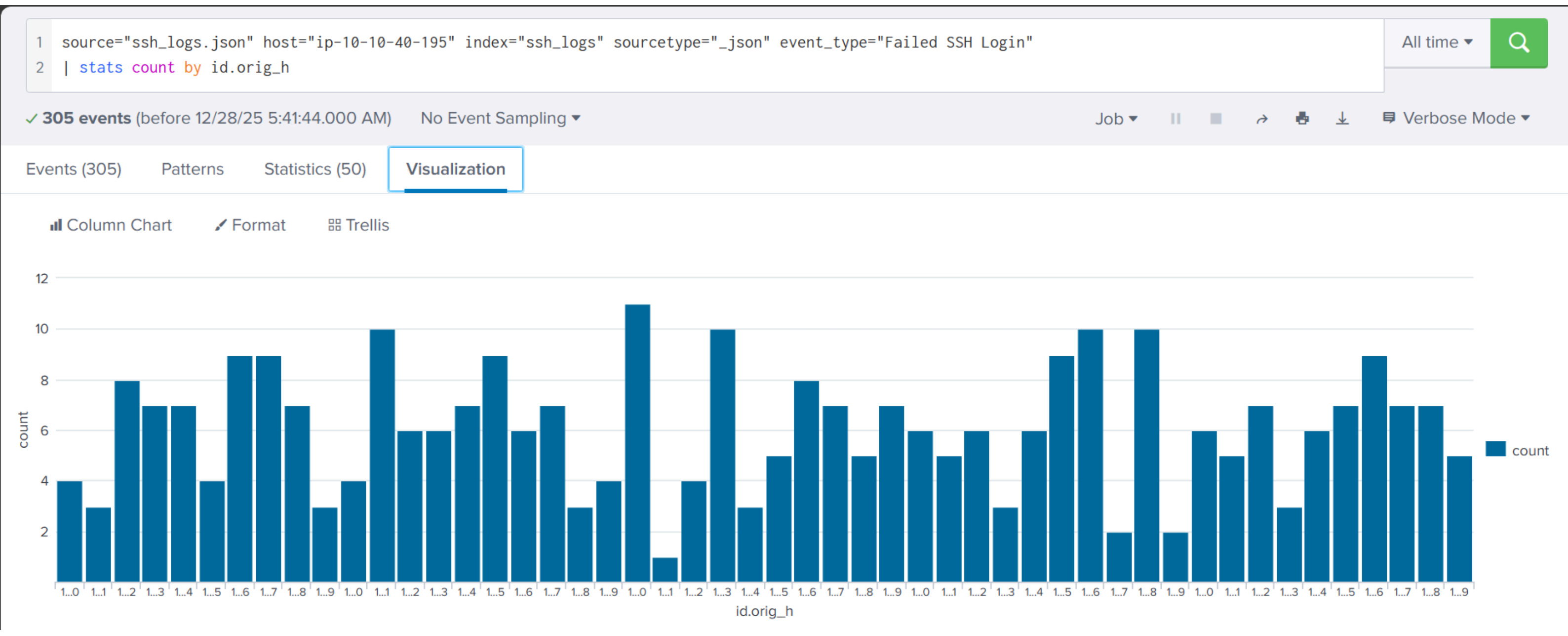
✓ **305 events** (before 12/28/25 6:00:50.000 AM) No Event Sampling

Events (305) Patterns Statistics (50) Visualization

100 Per Page Format Preview

id.orig_h	count
10.0.0.30	11
10.0.0.21	10
10.0.0.33	10
10.0.0.46	10
10.0.0.48	10
10.0.0.16	9
10.0.0.17	9
10.0.0.25	9
10.0.0.45	9
10.0.0.56	9

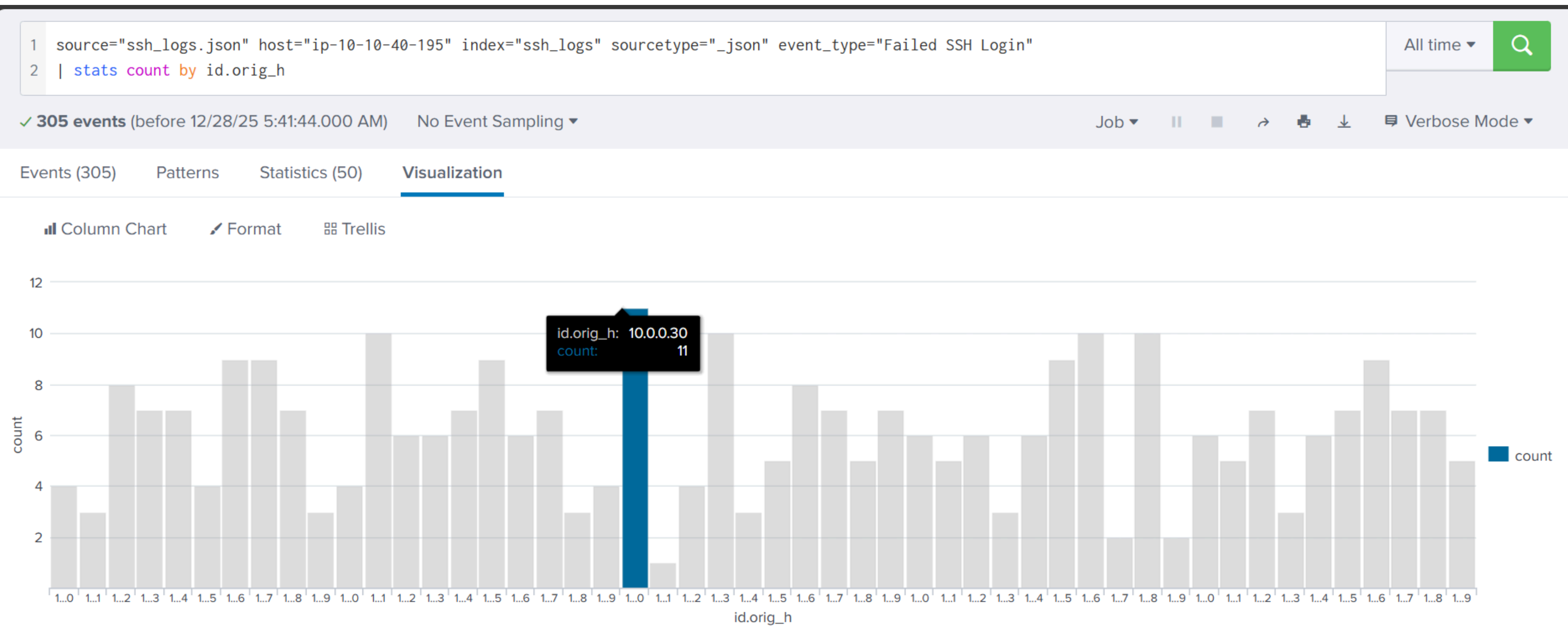
I created a bar chart visualization for failed attempts of multiple IP's



```
source="ssh_logs.json" host="ip-10-10-40-195" index="ssh_logs" sourcetype="_json" event_type="Failed SSH Login"
| stats count by id.orig_h
```

✓ 305 events (before 12/28/25 5:41:44.000 AM) No Event Sampling

There is an IP (10.0.0.30) has higher multiple failed attempts than other IP.



```
source="ssh_logs.json" host="ip-10-10-40-195" index="ssh_logs" sourcetype="_json" event_type="Failed SSH Login"
| stats count by id.orig_h
```

✓ 305 events (before 12/28/25 5:41:44.000 AM) No Event Sampling

id.orig_h: 10.0.0.30
count: 11

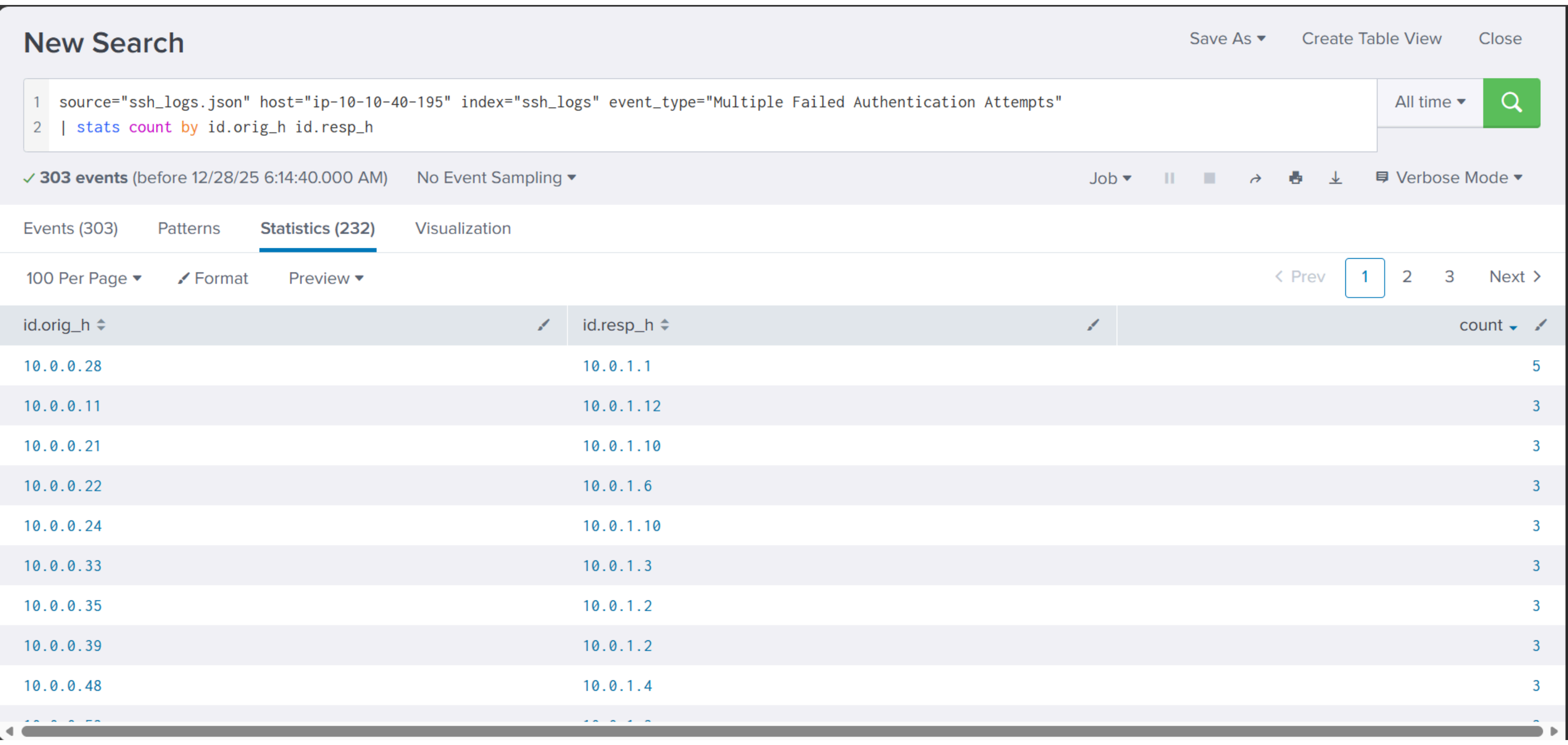
STEP - 3: MULTIPLE FAILED AUTHENTICATION ATTEMPTS

QUERY:

event_type="Multiple Failed Authentication Attempts"

| stats count by id.orig_h id.resp_h

This query helps to search the multiple failed authentication attempts. While analysing the SSH log, I found a failed login attempts by single IP. But, in this I'm going to find the failed authentication by an IP to another IP. Typically refers to a Brute-Force attack. It can be detected by seeing the repeated failures in the IP.



```
source="ssh_logs.json" host="ip-10-10-40-195" index="ssh_logs" event_type="Multiple Failed Authentication Attempts"
| stats count by id.orig_h id.resp_h
```

✓ 303 events (before 12/28/25 6:14:40.000 AM) No Event Sampling

id.orig_h	id.resp_h	count
10.0.0.28	10.0.1.1	5
10.0.0.11	10.0.1.12	3
10.0.0.21	10.0.1.10	3
10.0.0.22	10.0.1.6	3
10.0.0.24	10.0.1.10	3
10.0.0.33	10.0.1.3	3
10.0.0.35	10.0.1.2	3
10.0.0.39	10.0.1.2	3
10.0.0.48	10.0.1.4	3

Here, the IP 10.0.0.28 has the multiple failed authentication attempts to the IP of 10.0.1.1.

I can also create an alert for this type of attack for future. we can create alert by going to:

save as > alert > give a name at the title bar > alert type as real-time for real time attacks > add action > add triggered to alert > save

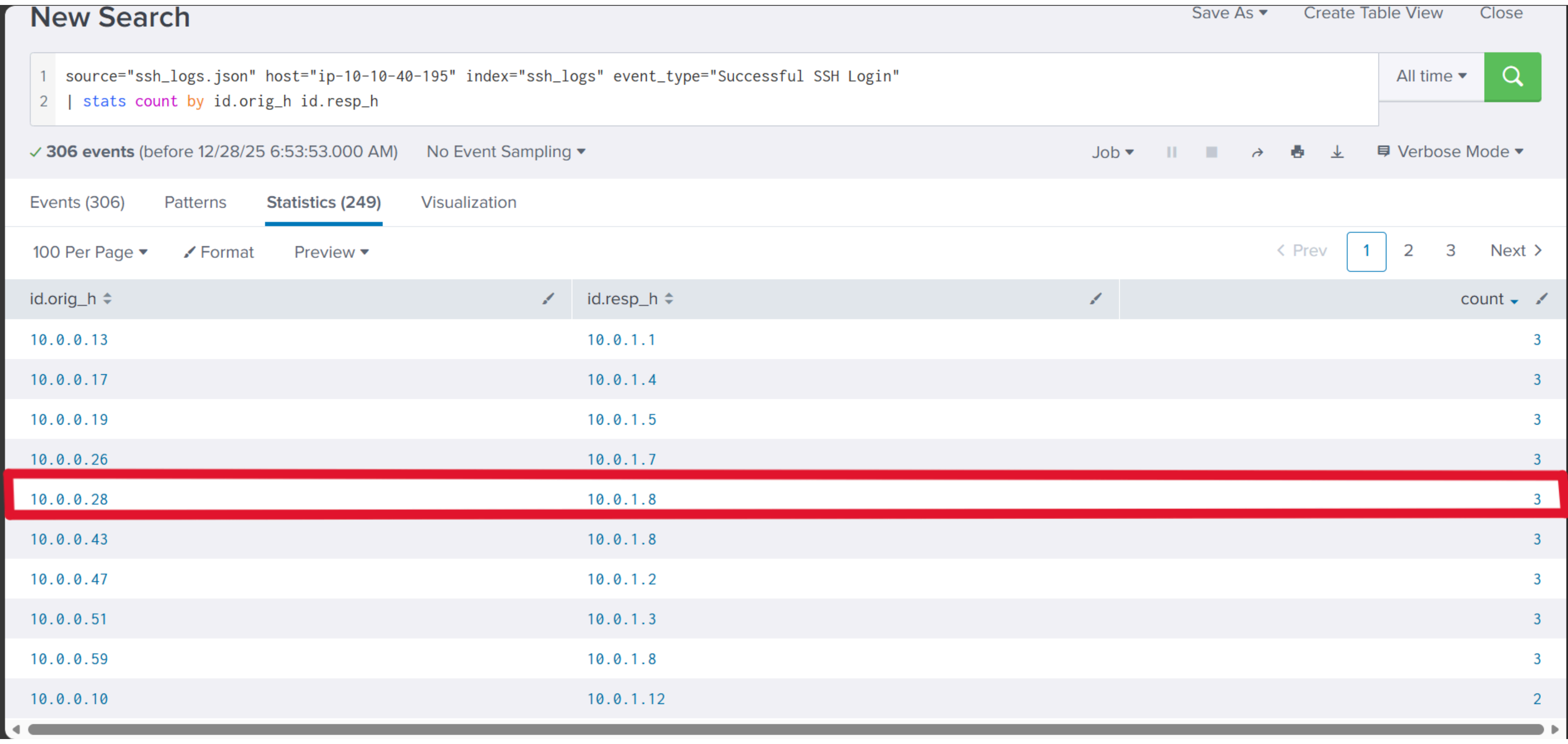
The alerts can be viewed at the activity section at the top of the page.

STEP - 4: TRACKING SUCCESSFUL LOGINS

Tracking the successful logins will helps us to detect that what if the brute-force attempted IP has successfully accessed. I can compare the IP 10.0.0.28 has successfully accessed to a machine.

query:

event_type="Successful SSH Login" | stats count by id.orig_h id.resp_h



```
source="ssh_logs.json" host="ip-10-10-40-195" index="ssh_logs" event_type="Successful SSH Login"
| stats count by id.orig_h id.resp_h
```

✓ 306 events (before 12/28/25 6:53:53.000 AM)

id.orig_h	id.resp_h	count
10.0.0.13	10.0.1.1	3
10.0.0.17	10.0.1.4	3
10.0.0.19	10.0.1.5	3
10.0.0.26	10.0.1.7	3
10.0.0.28	10.0.1.8	3
10.0.0.43	10.0.1.8	3
10.0.0.47	10.0.1.2	3
10.0.0.51	10.0.1.3	3
10.0.0.59	10.0.1.8	3
10.0.0.10	10.0.1.12	2

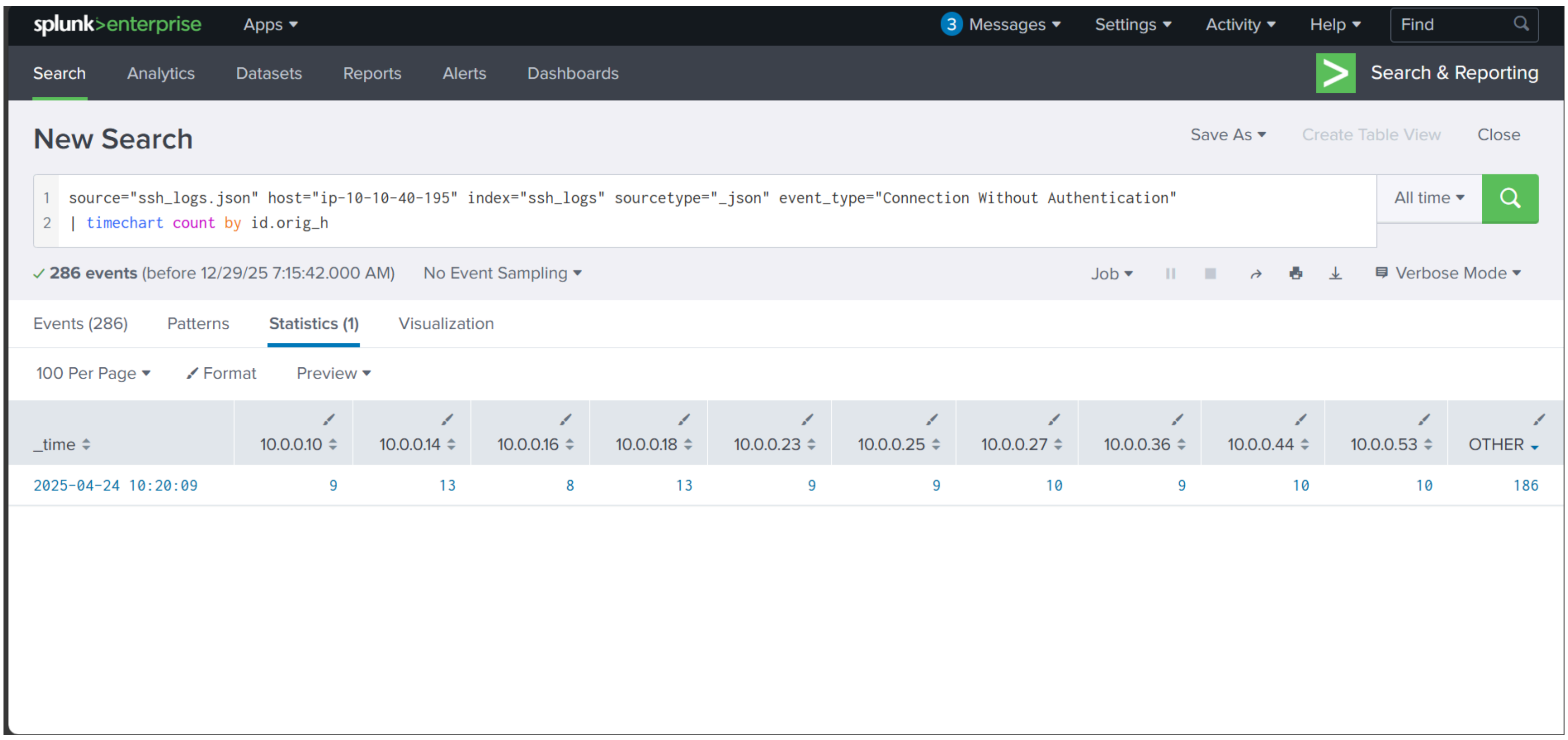
Here we can see that the IP 10.0.0.28 has as successful login attempt to 10.0.1.8.

STEP - 5: SUSPICIOUS CONNECTIONS WITIHOUT AUTHENTICATION

In this step, I’m going to create a time chart visualization for unauthenticated connection. This will help us to analyse which IP is done repeated authenticated attempts. Typically, refers to indication of port scanning or ssh probing.

query:

event_type="Connection Without Authentication" | timechart count by id.orig_h

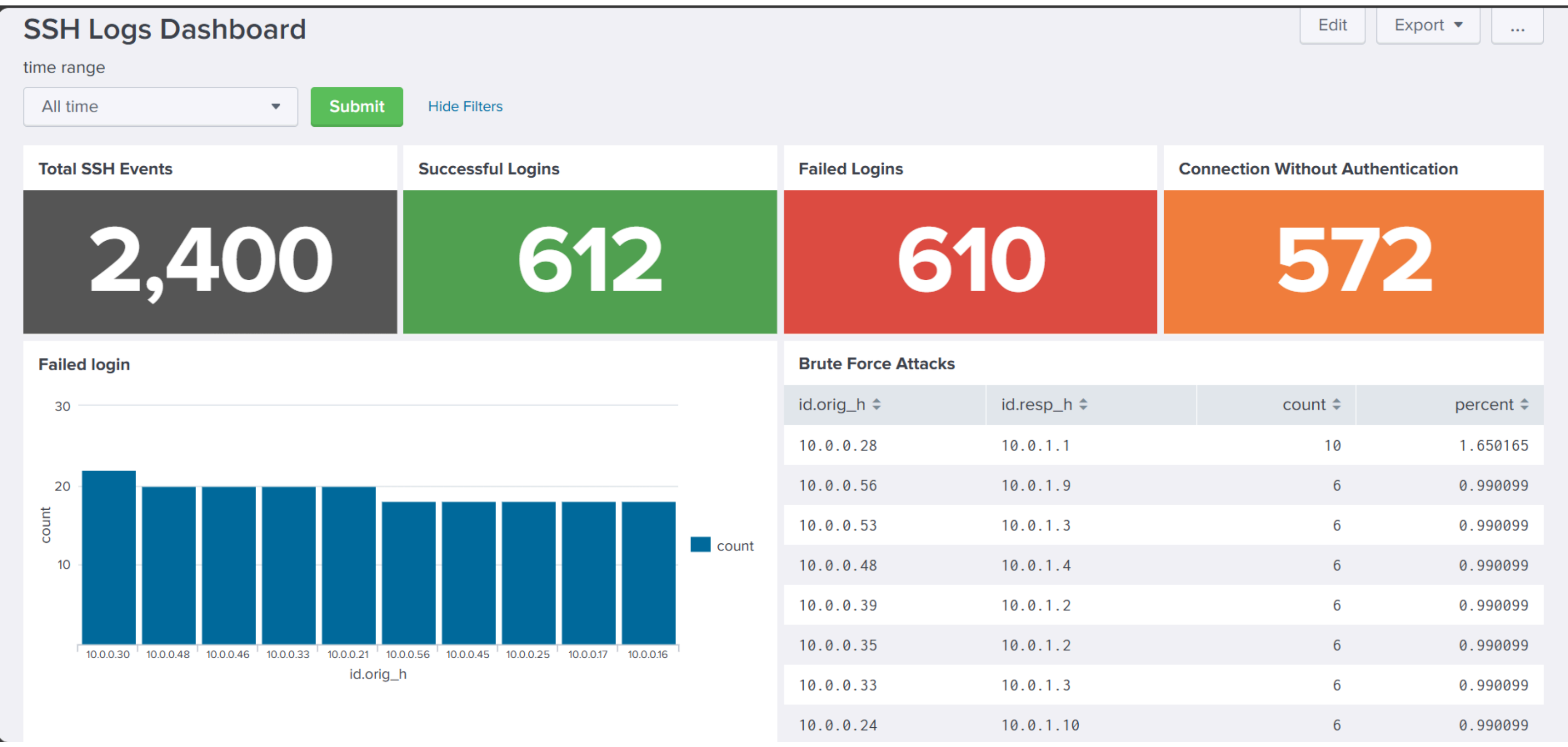


```
source="ssh_logs.json" host="ip-10-10-40-195" index="ssh_logs" sourcetype="_json" event_type="Connection Without Authentication"
| timechart count by id.orig_h
```

✓ 286 events (before 12/29/25 7:15:42.000 AM) No Event Sampling

_time	10.0.0.10	10.0.0.14	10.0.0.16	10.0.0.18	10.0.0.23	10.0.0.25	10.0.0.27	10.0.0.36	10.0.0.44	10.0.0.53	OTHER
2025-04-24 10:20:09	9	13	8	13	9	9	10	9	10	10	186

DASHBOARD



SSH Logs Dashboard

time range: All time

Total SSH Events: 2,400

Successful Logins: 612

Failed Logins: 610

Connection Without Authentication: 572

Failed login

Brute Force Attacks

id.orig_h	id.resp_h	count	percent
10.0.0.28	10.0.1.1	10	1.650165
10.0.0.56	10.0.1.9	6	0.990099
10.0.0.53	10.0.1.3	6	0.990099
10.0.0.48	10.0.1.4	6	0.990099
10.0.0.39	10.0.1.2	6	0.990099
10.0.0.35	10.0.1.2	6	0.990099
10.0.0.33	10.0.1.3	6	0.990099
10.0.0.24	10.0.1.10	6	0.990099

MITIGATION

1. Failed Login Attempts

- Enforce account lockout policies after a defined number of failed attempts
- Implement strong password policies (length, complexity, rotation where appropriate)
- Enable Multi-Factor Authentication (MFA) for all users, especially privileged accounts

2. Multiple Failed Authentication Attempts

- Enforce progressive account lockouts or temporary blocks
- Enable adaptive authentication (risk-based login controls)
- Apply IP reputation filtering and geo-blocking where applicable
- Use Web Application Firewalls (WAFs) to block automated attacks

3. Successful Logins After Failed Attempts

- Immediately **flag and investigate** accounts with this pattern
- Force **password reset** for affected accounts
- Require **step-up authentication** (MFA re-verification)
- Terminate active sessions if suspicious behavior is confirmed

4. Suspicious Connections Without Authentication

- Monitor for:
    - Port scanning
    - Enumeration attempts
- Apply IDS/IPS rules to detect anomalous traffic
- Log and alert on all denied or unauthenticated connection attempts
- Conduct regular attack surface reviews and vulnerability scans

HTTP LOG ANALYSIS

HTTP logs are records generated by web servers, web applications, proxies, and web application firewalls that capture details of HTTP requests and responses between clients and servers. They typically include information such as the client IP address, timestamp, HTTP method (GET, POST, PUT, DELETE), requested URL, response status code, bytes transferred, referrer, and user-agent. HTTP logs are essential for monitoring website activity, troubleshooting errors, analyzing performance, and detecting security threats. By reviewing HTTP logs, organizations can identify suspicious behavior such as repeated failed login attempts, brute-force attacks, unauthorized access, malicious payloads, and abnormal traffic patterns, making them a critical data source for security monitoring, incident response, and compliance auditing.

By analyzing HTTP logs, security teams can identify suspicious activities such as brute-force login attempts, unauthorized access, SQL injection, cross-site scripting (XSS), directory traversal, and connections without authentication. HTTP log analysis is commonly performed using SIEM tools like Splunk to correlate events, generate alerts, support forensic investigations, and ensure compliance with security and auditing requirements.

PROCEDURE

STEP - 1: TOP 10 ENDPOINTS THAT GENERATING WEB TRAFFIC

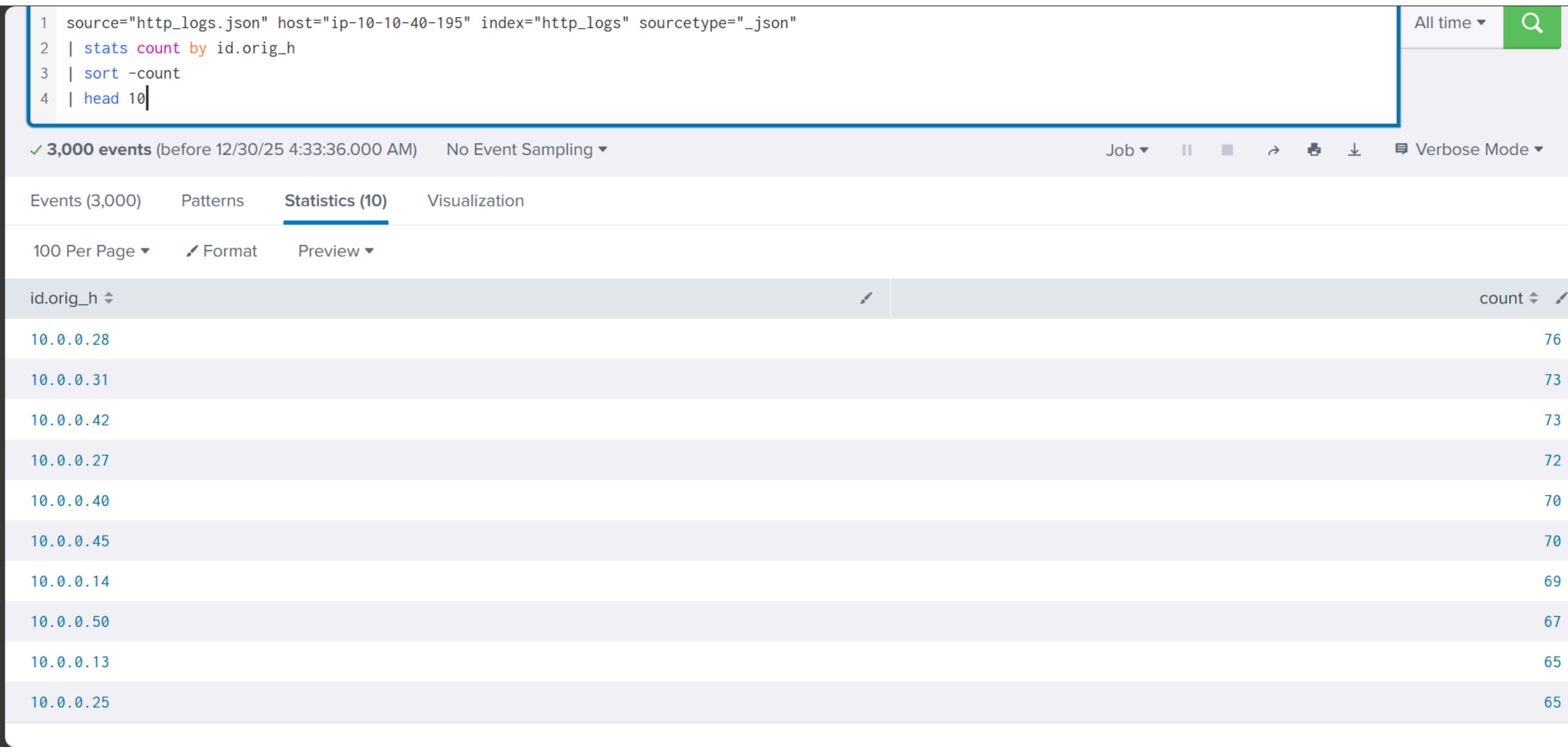
query:

index="http_logs" | stats count by id.orig_h | sort -count | head 10

This command helps to see the top 10 IP’s of an endpoint device from the index http_logs.

The command **sort -count** will sort result by count in reverse order.

The **head 10** command will display top 10 IP.



```
source="http_logs.json" host="ip-10-10-40-195" index="http_logs" sourcetype="_json"
| stats count by id.orig_h
| sort -count
| head 10
```

✓ 3,000 events (before 12/30/25 4:33:36.000 AM) No Event Sampling

Events (3,000) Patterns Statistics (10) Visualization

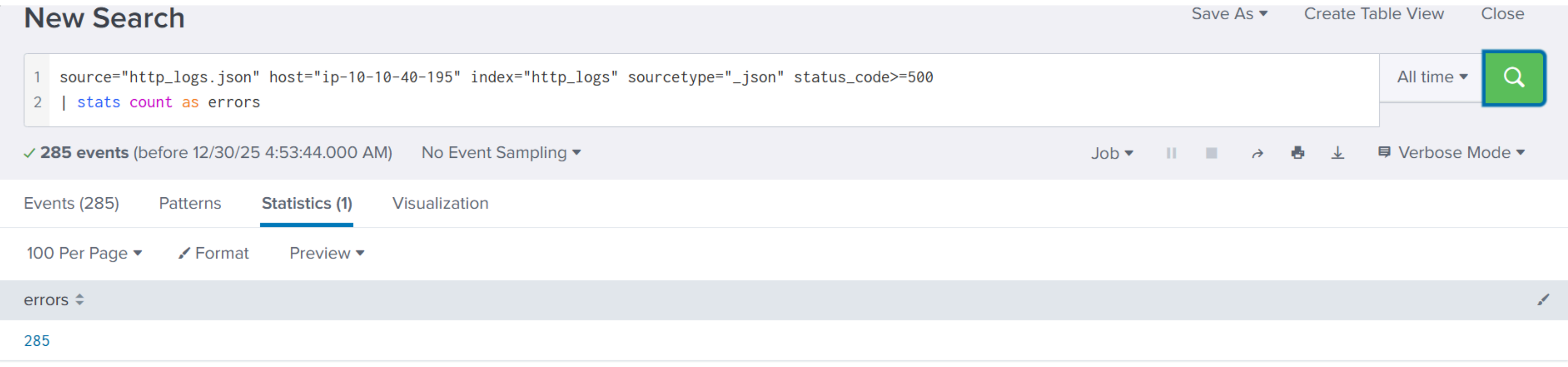
id.orig_h	count
10.0.0.28	76
10.0.0.31	73
10.0.0.42	73
10.0.0.27	72
10.0.0.40	70
10.0.0.45	70
10.0.0.14	69
10.0.0.50	67
10.0.0.13	65
10.0.0.25	65

STEP - 2: NUMBER OF SERVER ERRORS

query:

index="http_logs" status_code>=500 | stats count as errors

With this command, I can check the number of server errors. The status code for server errors is 500. So, I can search for the status code greater than 500. And the command **stats count as errors** replaces the name count as errors for better understanding.



```
source="http_logs.json" host="ip-10-10-40-195" index="http_logs" sourcetype="_json" status_code>=500
| stats count as errors
```

✓ 285 events (before 12/30/25 4:53:44.000 AM) No Event Sampling

errors
285

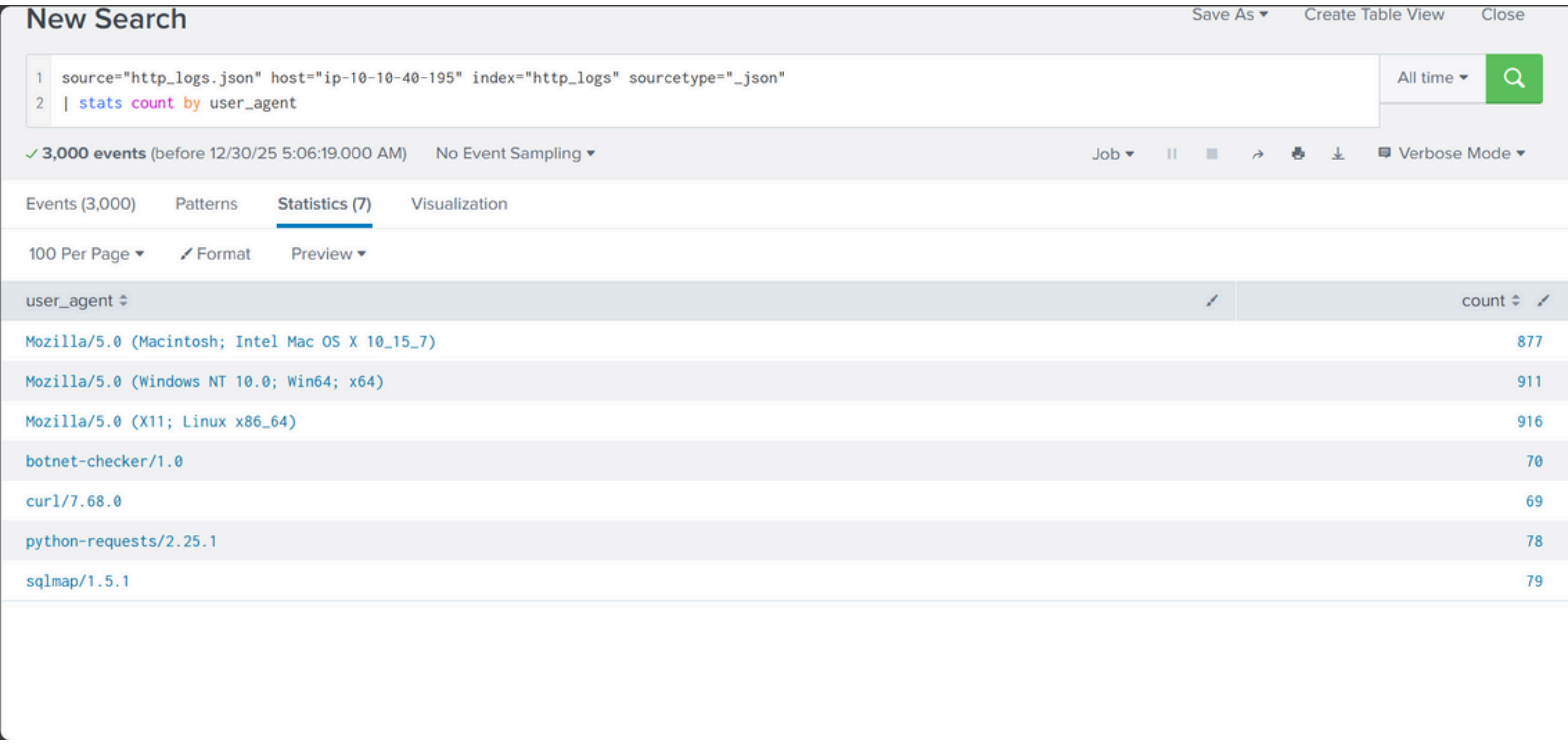
Here, the results show the error count as 285. So, there are 285 server errors.

STEP - 3: POSSIBLE SCRIPTED ATTACKS:

To check the possible scripted attacks, user_agent field helps us to see the scripted attacks. user_agent field is known as browser where the user uses.

query:

stats count by user_agent



```
source="http_logs.json" host="ip-10-10-40-195" index="http_logs" sourcetype="_json"
| stats count by user_agent
```

✓ 3,000 events (before 12/30/25 5:06:19.000 AM) No Event Sampling

user_agent	count
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)	877
Mozilla/5.0 (Windows NT 10.0; Win64; x64)	911
Mozilla/5.0 (X11; Linux x86_64)	916
botnet-checker/1.0	70
curl/7.68.0	69
python-requests/2.25.1	78
sqlmap/1.5.1	79

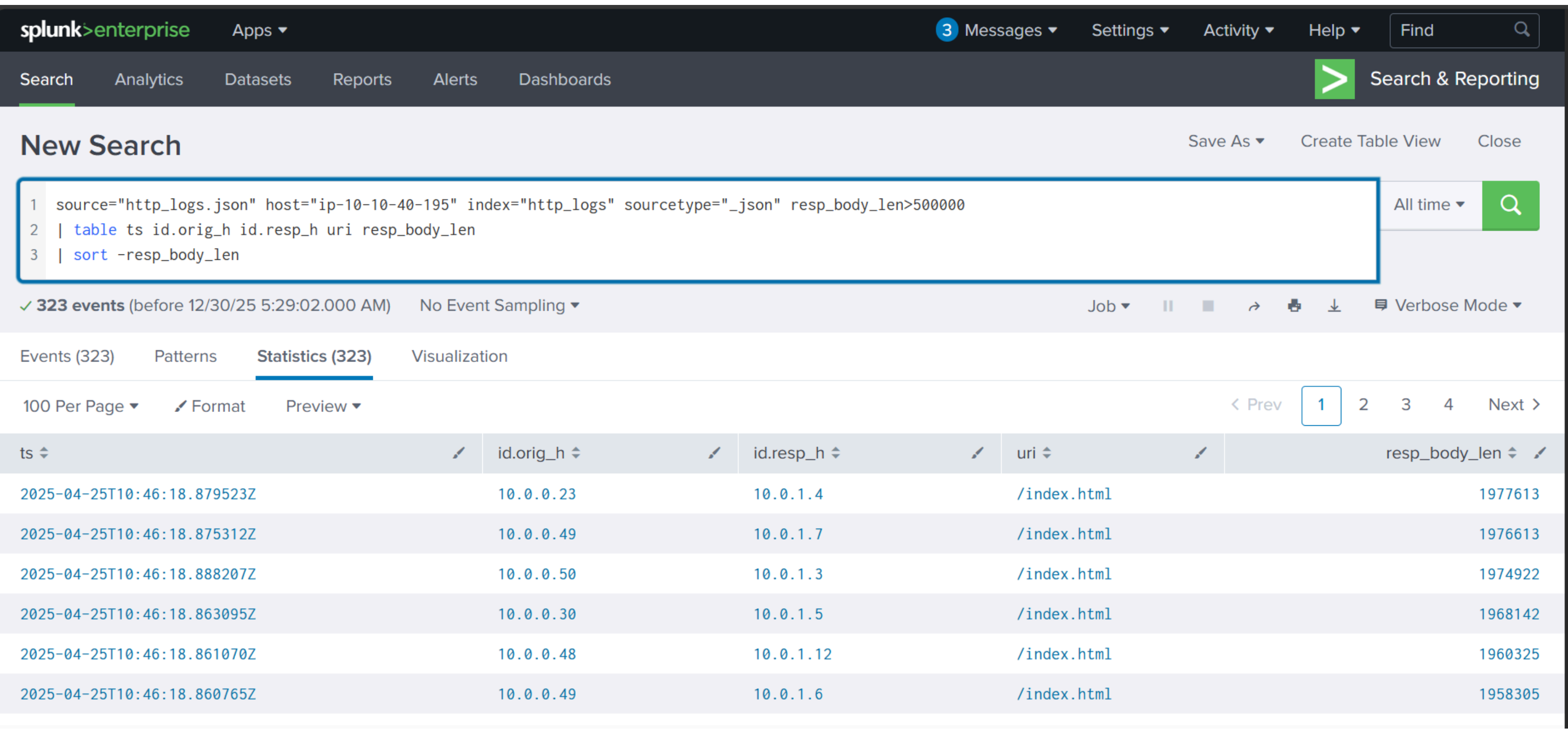
Here, I can see the number of user_agents in this log. Mozilla browser are normal browsers, where user uses, but curl, python-requests, sqlmap are a possible sign of scripted attacks. Curl, python-requests, sqlmap are not normally used by the users.

STEP - 4: LARGE FILE TRANSFERS

query:

resp_body_len>500000 | table ts id.orig_h id.resp_h uri resp_body_len | sort -resp_body_len

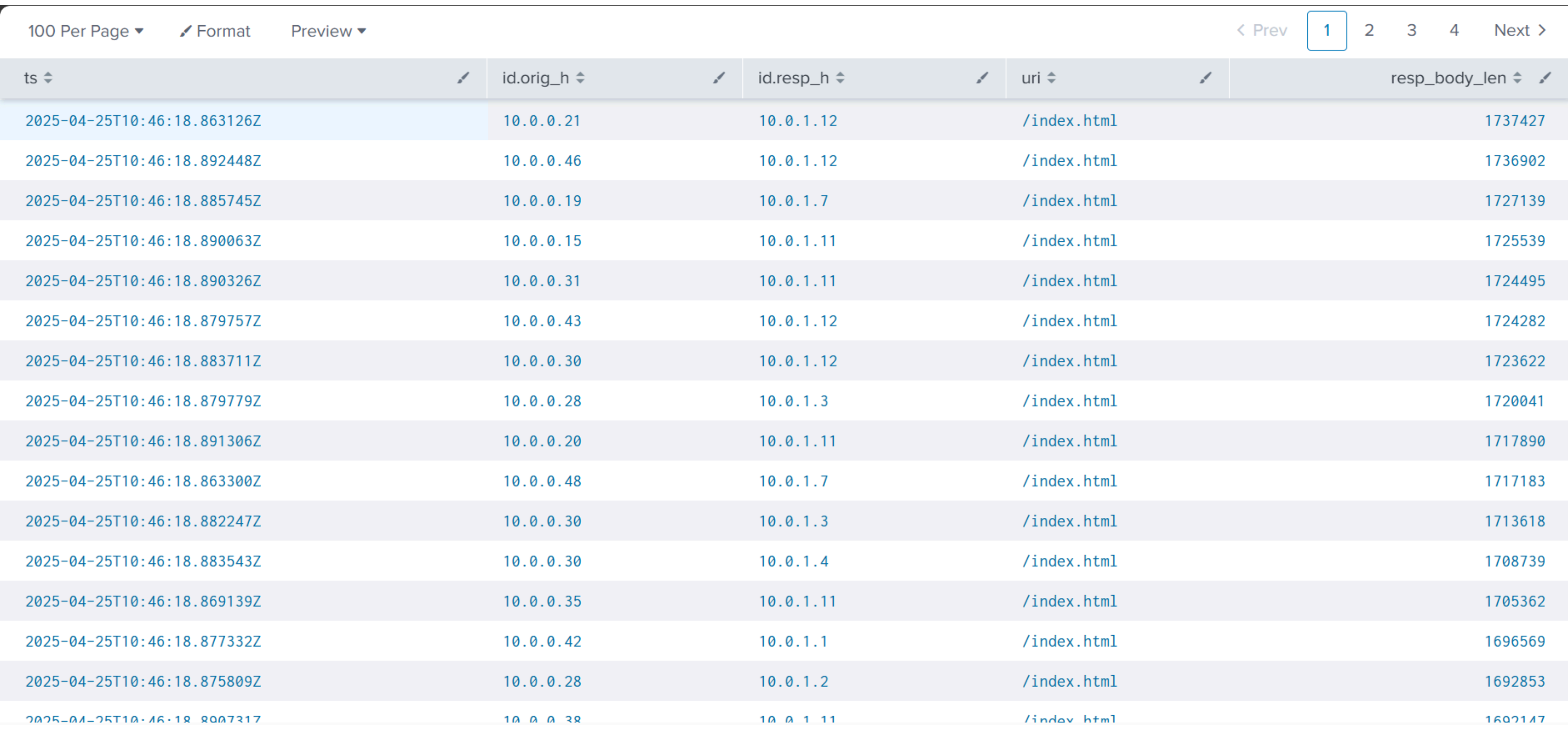
This query helps us to view the body length of file transfer happened in the log. Here, I took the file transfer limit as greater than 500kb which is 500000 bytes. The **ts, id.orig_h, id.resp_h, uri, resp_body_len** are the timestamp, source Ip, destination Ip, and body length of the file transfer. The sort -resp_body_len is sorting the body length of the file transfer from higher to lower.



```
source="http_logs.json" host="ip-10-10-40-195" index="http_logs" sourcetype="_json" resp_body_len>500000
| table ts id.orig_h id.resp_h uri resp_body_len
| sort -resp_body_len
```

✓ 323 events (before 12/30/25 5:29:02.000 AM)   No Event Sampling

ts	id.orig_h	id.resp_h	uri	resp_body_len
2025-04-25T10:46:18.879523Z	10.0.0.23	10.0.1.4	/index.html	1977613
2025-04-25T10:46:18.875312Z	10.0.0.49	10.0.1.7	/index.html	1976613
2025-04-25T10:46:18.888207Z	10.0.0.50	10.0.1.3	/index.html	1974922
2025-04-25T10:46:18.863095Z	10.0.0.30	10.0.1.5	/index.html	1968142
2025-04-25T10:46:18.861070Z	10.0.0.48	10.0.1.12	/index.html	1960325
2025-04-25T10:46:18.860765Z	10.0.0.49	10.0.1.6	/index.html	1958305



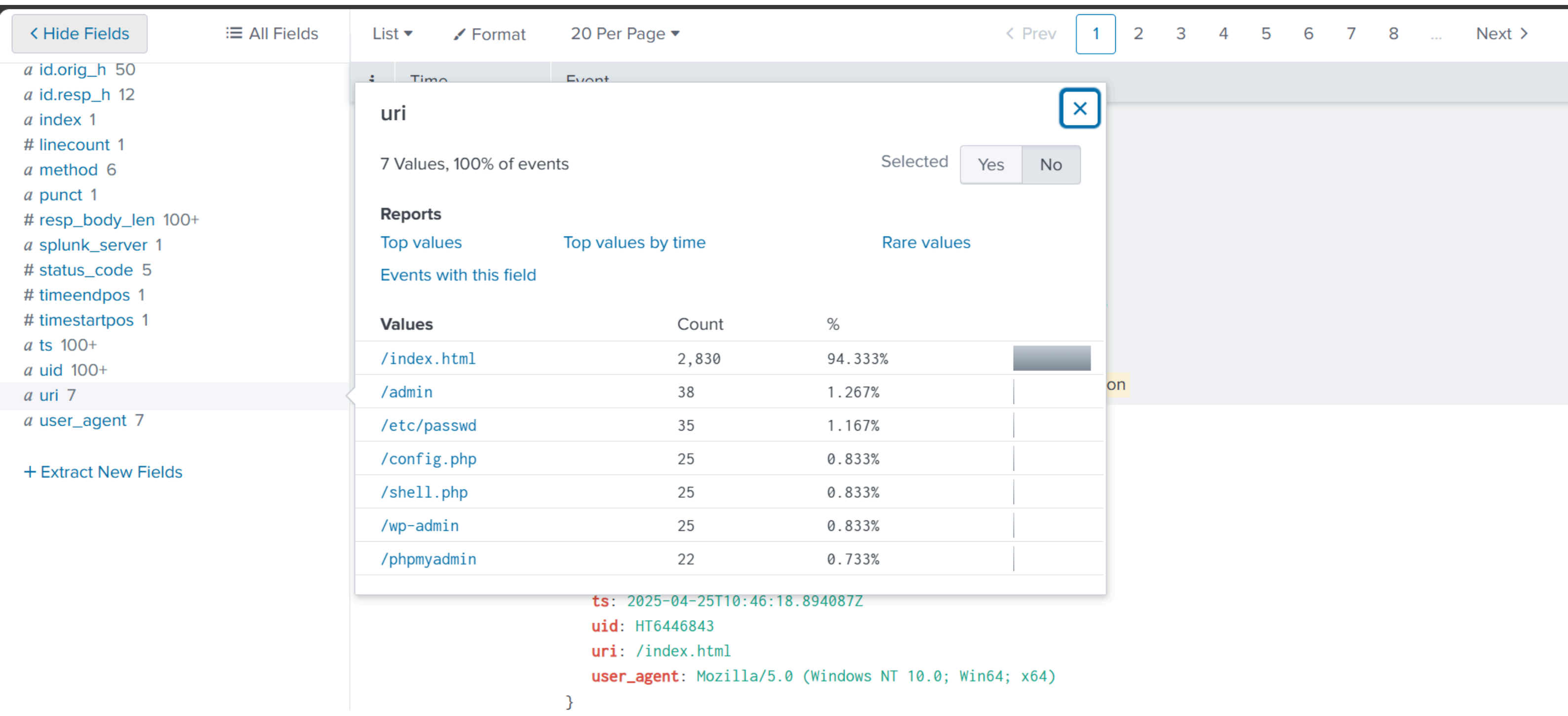
ts	id.orig_h	id.resp_h	uri	resp_body_len
2025-04-25T10:46:18.863126Z	10.0.0.21	10.0.1.12	/index.html	1737427
2025-04-25T10:46:18.892448Z	10.0.0.46	10.0.1.12	/index.html	1736902
2025-04-25T10:46:18.885745Z	10.0.0.19	10.0.1.7	/index.html	1727139
2025-04-25T10:46:18.890063Z	10.0.0.15	10.0.1.11	/index.html	1725539
2025-04-25T10:46:18.890326Z	10.0.0.31	10.0.1.11	/index.html	1724495
2025-04-25T10:46:18.879757Z	10.0.0.43	10.0.1.12	/index.html	1724282
2025-04-25T10:46:18.883711Z	10.0.0.30	10.0.1.12	/index.html	1723622
2025-04-25T10:46:18.879779Z	10.0.0.28	10.0.1.3	/index.html	1720041
2025-04-25T10:46:18.891306Z	10.0.0.20	10.0.1.11	/index.html	1717890
2025-04-25T10:46:18.863300Z	10.0.0.48	10.0.1.7	/index.html	1717183
2025-04-25T10:46:18.882247Z	10.0.0.30	10.0.1.3	/index.html	1713618
2025-04-25T10:46:18.883543Z	10.0.0.30	10.0.1.4	/index.html	1708739
2025-04-25T10:46:18.869139Z	10.0.0.35	10.0.1.11	/index.html	1705362
2025-04-25T10:46:18.877332Z	10.0.0.42	10.0.1.1	/index.html	1696569
2025-04-25T10:46:18.875809Z	10.0.0.28	10.0.1.2	/index.html	1692853
2025-04-25T10:46:18.890731Z	10.0.0.38	10.0.1.11	/index.html	1692147

STEP - 5: DETECT SUSPICIOUS URL'S ACCESSED

query:

uri IN (/admin, /etc/passwd, /shell.php) | stats count by uri id.orig_h

In this step, I'm detecting the suspicious url's accessed by users. To check what url's accessed, I can use the **uri** field in the events section.

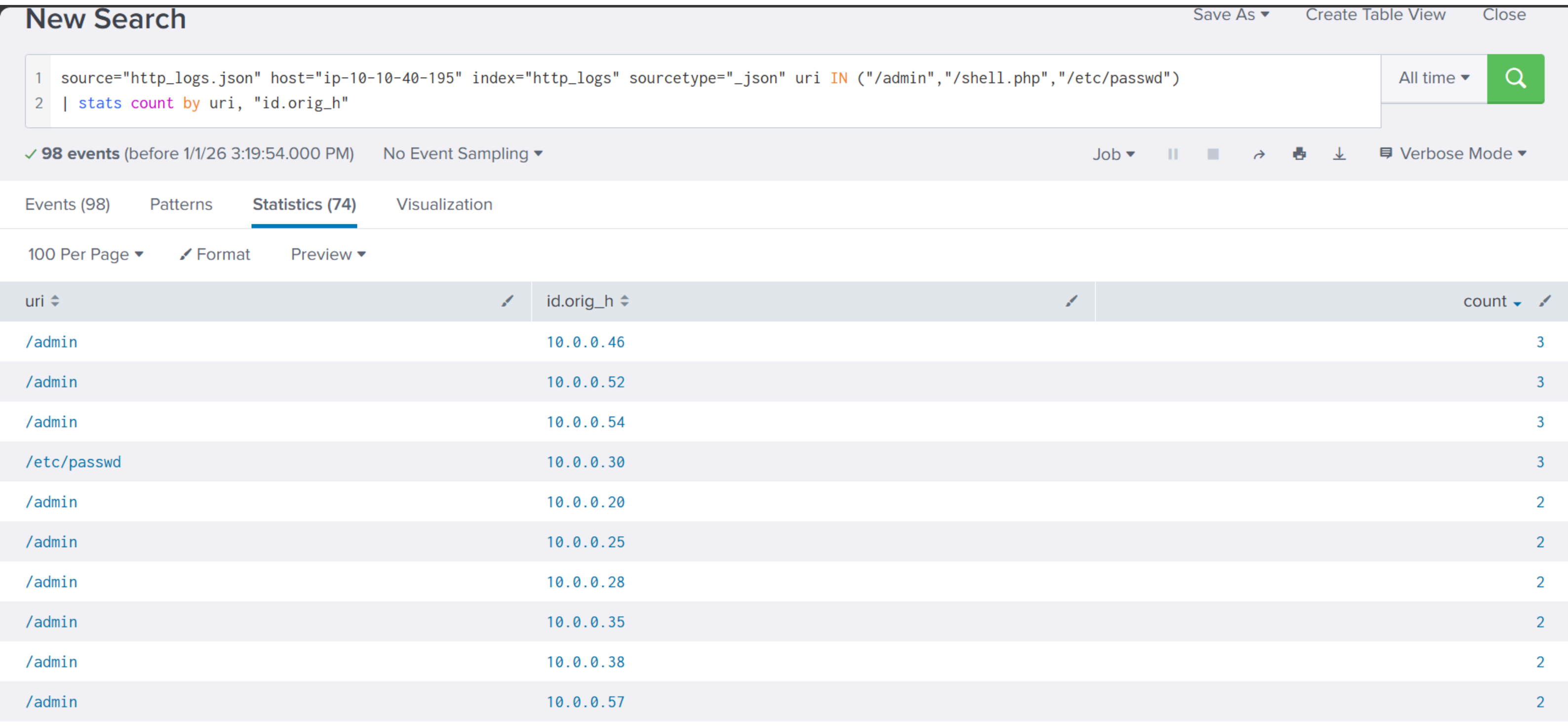


Values	Count	%
/index.html	2,830	94.333%
/admin	38	1.267%
/etc/passwd	35	1.167%
/config.php	25	0.833%
/shell.php	25	0.833%
/wp-admin	25	0.833%
/phpmyadmin	22	0.733%

```
ts: 2025-04-25T10:46:18.894087Z
uid: HT6446843
uri: /index.html
user_agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
}
```

Here, I can see the list of URLs accessed in this log. Now, I have to filter out the suspicious URL's and find the source Ip that accessed these URLs. **/admin, /etc/passwd, /shell.php** are the URLs we need.

This is where I used the above query. Using the **uri** field and adding the URLs I need and **stats count by uri** and **id.orig_h** (source Ip), I can see the URLs accessed by source Ip.

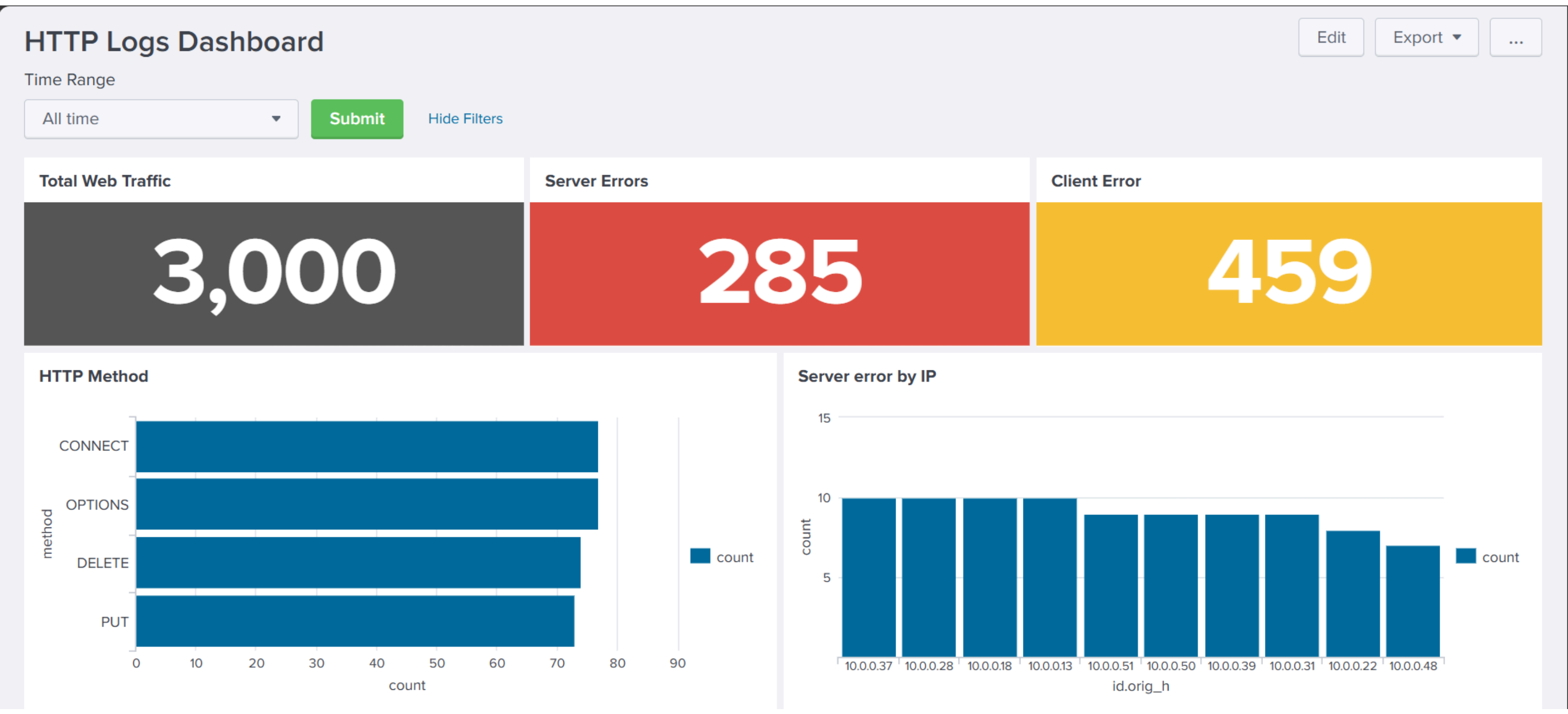


```
source="http_logs.json" host="ip-10-10-40-195" index="http_logs" sourcetype="_json" uri IN ("/admin","/shell.php","/etc/passwd")
| stats count by uri, "id.orig_h"
```

✓ **98 events** (before 1/1/26 3:19:54.000 PM)

uri	id.orig_h	count
/admin	10.0.0.46	3
/admin	10.0.0.52	3
/admin	10.0.0.54	3
/etc/passwd	10.0.0.30	3
/admin	10.0.0.20	2
/admin	10.0.0.25	2
/admin	10.0.0.28	2
/admin	10.0.0.35	2
/admin	10.0.0.38	2
/admin	10.0.0.57	2

DASHBOARD



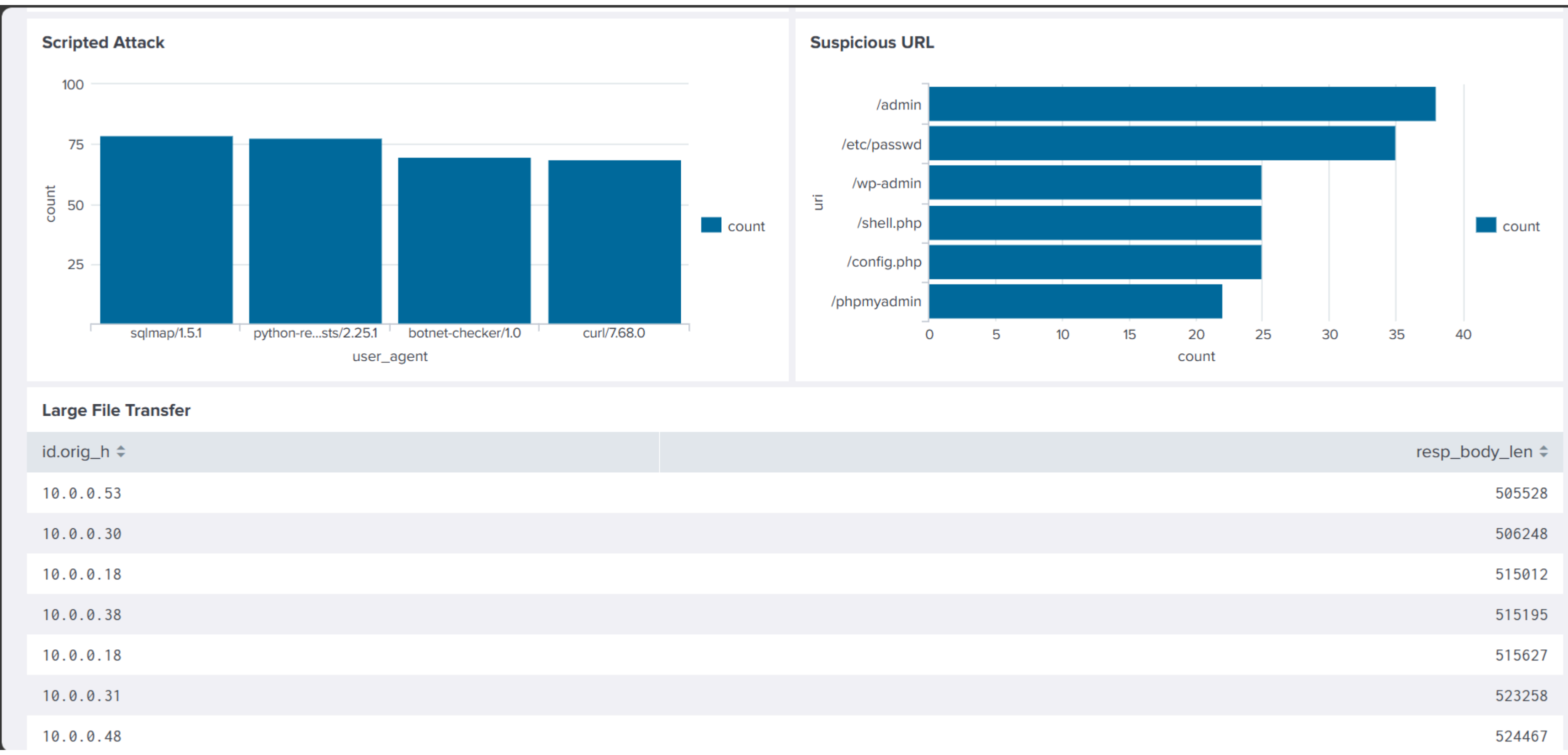
HTTP Logs Dashboard

Time Range: All time

Total Web Traffic: 3,000

Server Errors: 285

Client Error: 459



Large File Transfer

id.orig_h	resp_body_len
10.0.0.53	505528
10.0.0.30	506248
10.0.0.18	515012
10.0.0.38	515195
10.0.0.18	515627
10.0.0.31	523258
10.0.0.48	524467

MITIGATION

1) ENDPOINTS THAT GENERATING WEB TRAFFIC

- **DNS monitoring:**
  - Detect DGA domains, excessive NXDOMAIN responses, or newly registered domains.
- **Endpoint inspection:**
  - Scan endpoints with high traffic for malware, browser extensions, or rogue scripts.
- **Proxy enforcement:**
  - Force all HTTP/HTTPS traffic through a secure web gateway (SWG).

2) High Number of Server Errors

- **Error-rate alerts**:
  - Trigger alerts on spikes in error responses per source.
- **Harden error handling**:
  - Suppress verbose error messages (no stack traces or system info).
- **Patch and config review**:
  - Ensure web servers, frameworks, and plugins are up to date.
- **Credential protection**:
  - Apply CAPTCHA or MFA when repeated auth failures are detected.

3) POSSIBLE SCRIPTED ATTACKS

- **User-Agent filtering**:
  - Block or challenge requests with known automation UAs unless explicitly required.
- **Bot management**:
  - Use bot-detection features in WAF/CDN (JavaScript challenges, fingerprinting).
- **Allowlist legitimate automation**:

- Document and allow known CI/CD, monitoring, or API clients.

4) Large File Transfers

- **DLP controls**:
    - Inspect outbound transfers for sensitive data patterns.
- **Transfer size thresholds**:
    - Alert on uploads/downloads exceeding normal size per endpoint or app.
- **Protocol restrictions**:
    - Limit or block FTP, SCP, or unauthorized cloud storage services.

5) Access to Suspicious URIs

- **Immediate blocking**:
    - Block requests matching known exploit paths via WAF.
- **Deception responses**:
    - Return generic 404s or tarpits instead of informative errors.
- **Rate limiting & IP reputation**:
    - Temporarily block IPs scanning multiple sensitive paths.

CONCLUSION

This project demonstrated a basic approach to log analysis using Splunk by examining SSH and HTTP logs to identify common security-related events such as failed authentication attempts, multiple login failures, successful logins, unauthenticated connections, and suspicious web activity. By ingesting logs into a SIEM platform and applying simple SPL queries, the project showed how raw log data can be filtered, summarized, and visualized to highlight potential security issues.

The analysis focused on identifying patterns that may indicate brute-force attacks, scanning behavior, scripted web requests, server errors, and access to sensitive URLs. Basic alert creation and visualizations were used to simulate how such events might be monitored in a SOC environment. The project also outlined general mitigation steps to address the identified issues, emphasizing standard security controls such as MFA, account lockouts, WAF rules, and monitoring.

Overall, this project serves as a learning exercise to understand how log data is used within a SIEM for security monitoring and investigation. While the scope is limited and simplified compared to real-world SOC operations, it provides foundational exposure to log ingestion, search queries, basic detection logic, and security analysis concepts that are relevant to entry-level cybersecurity and SOC roles.