HTTP Security Analysis with Splunk

Date: 2025-08-17

Analyst: Vlassov Vladislav

Data Source: Splunk sourcetype="access\_combined"

# Executive Summary

This portfolio documents six HTTP-focused Splunk searches with paired screenshots, covering status trends, error rates, brute-force on /login, SQLi scanning, traversal/CGI probing, and webshell indicators.

# Methodology

Each section includes the SPL query, result, analysis, and conclusion. Screenshots follow the order of the source document for 1:1 mapping.

## Key Fields

|  |  |
| --- | --- |
| Field | Description |
| \_time | Event timestamp |
| clientip | Client source IP |
| method | HTTP method |
| uri | Requested path and query string |
| status | HTTP status code |
| useragent | Client User-Agent |
| bytes | Response size (bytes) |
| referer | Referrer header |

# 1. Status-Class Trend Over Time

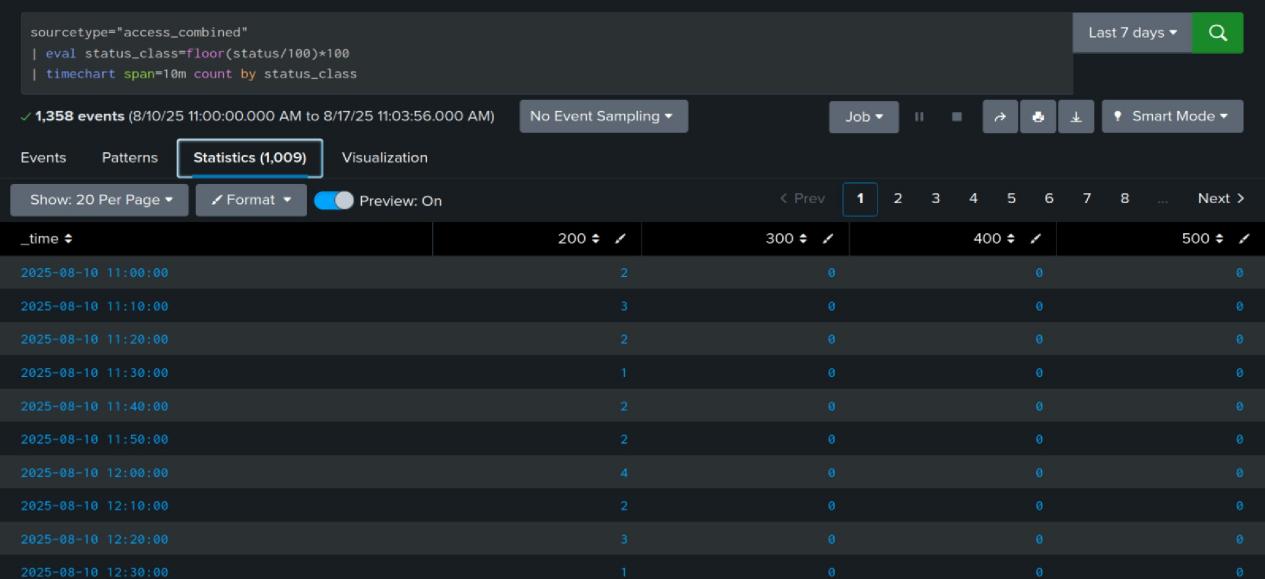
## Objective

Establish baseline and detect spikes across 2xx/3xx/4xx/5xx.

## Splunk Query

sourcetype="access\_combined" | eval status\_class=floor(status/100)\*100 | timechart span=10m count by status\_class

## Result



## Analysis

Baseline status mix should be dominated by 2xx/3xx. Spikes in 4xx/5xx indicate client abuse or backend issues.

## Conclusion

Baseline established; anomalous periods isolated for review.

# 2. 5xx Error Rate Over Time

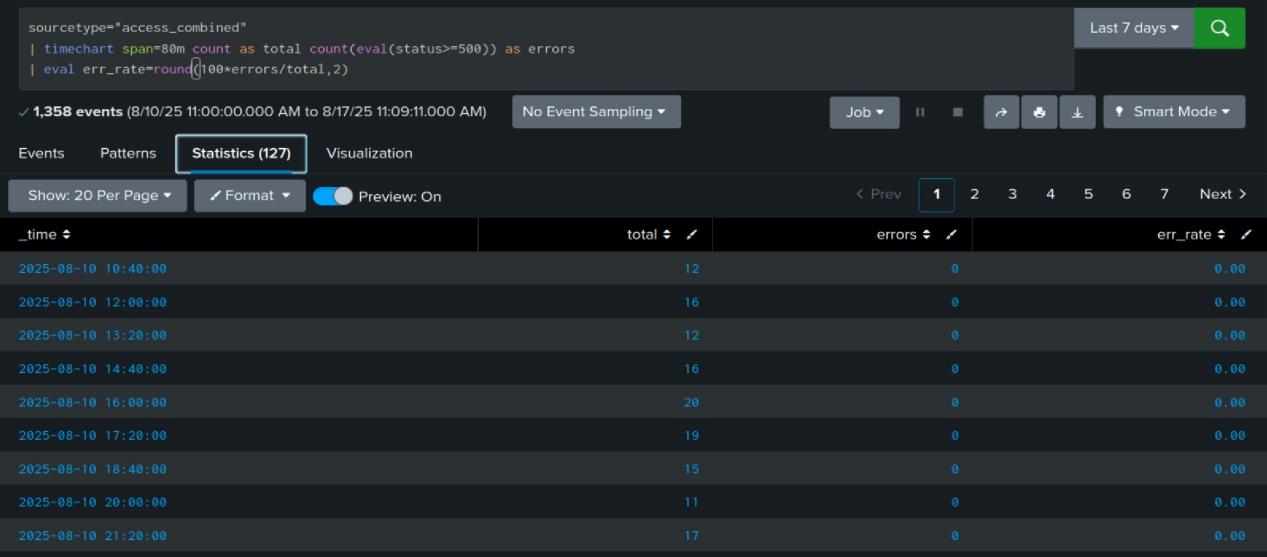
## Objective

Track total traffic and server errors to compute error rate.

## Splunk Query

sourcetype="access\_combined" | timechart span=80m count as total count(eval(status>=500)) as errors | eval err\_rate=round(100\*errors/total,2)

## Result



## Analysis

Sudden error-rate jumps often align with deploys, outages, or attacks. Compare with change windows.

## Conclusion

Error-rate anomalies identified; correlate with releases and incidents.

# 3. Login Brute-Force by Client IP

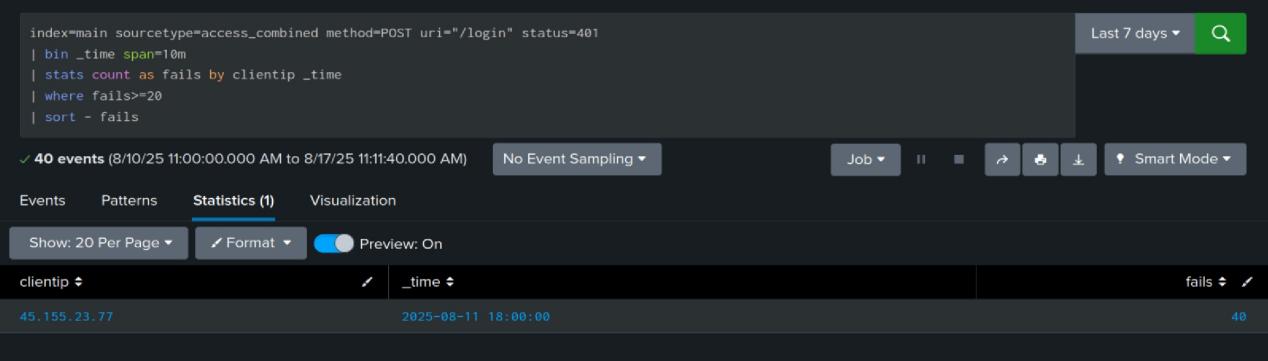
## Objective

Detect excessive failed POST /login attempts per client within 10-minute bins.

## Splunk Query

index=main sourcetype=access\_combined method=POST uri="/login" status=401 | bin \_time span=10m | stats count as fails by clientip \_time | where fails>=20 | sort - fails

## Result



## Analysis

High fail counts from the same IP strongly suggest brute-force. Geo/ASN context helps prioritization.

## Conclusion

Brute-force sources prioritized for blocking and credential protections.

# 4. SQL Injection Scanning (sqlmap and payloads)

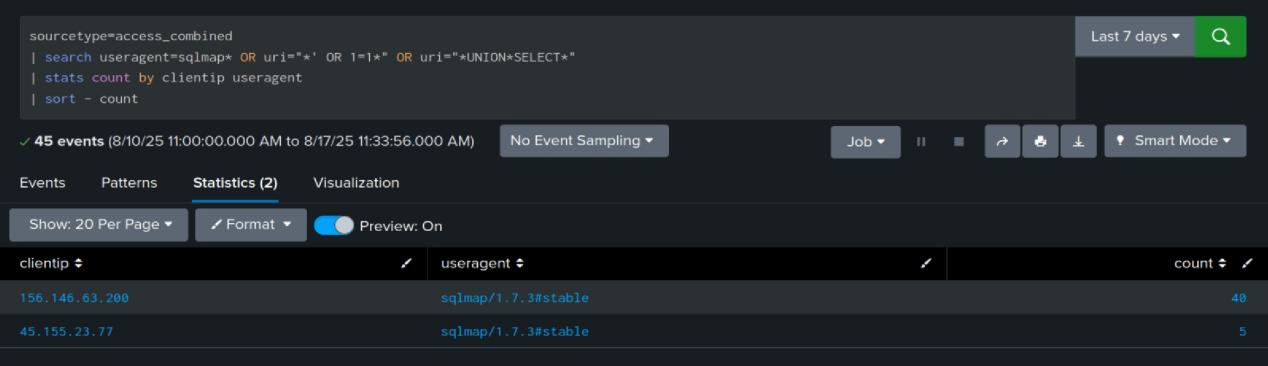
## Objective

Identify automated SQL injection scanners via user-agent and payloads.

## Splunk Query

sourcetype=access\_combined | search useragent=sqlmap\* OR uri="\*' OR 1=1\*" OR uri="\*UNION\*SELECT\*" | stats count by clientip useragent | sort - count

## Result



## Analysis

sqlmap UAs and classic injection payloads are high-confidence scanner signals. Review WAF blocks.

## Conclusion

Injection scanning sources enumerated; tighten WAF and rate limits.

# 5. Path Traversal and CGI Scanning

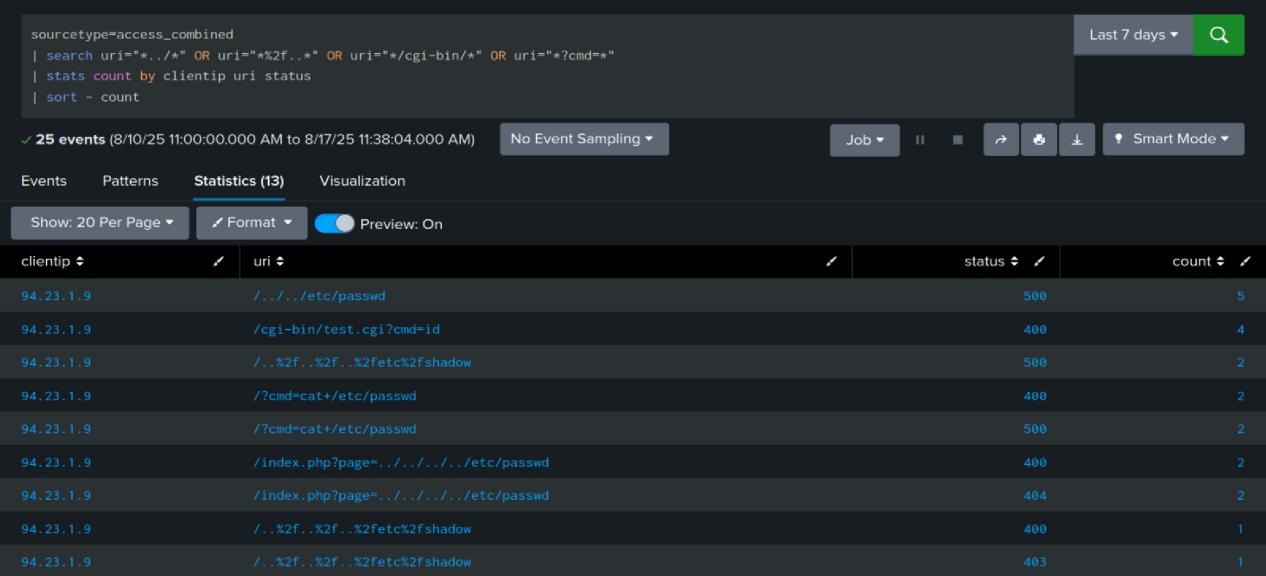
## Objective

Catch traversal and CGI probing attempts via URI patterns.

## Splunk Query

sourcetype=access\_combined | search uri="\*../\*" OR uri="\*%2f..\*" OR uri="\*/cgi-bin/\*" OR uri="\*?cmd=\*" | stats count by clientip uri status | sort - count

## Result



## Analysis

Traversal markers and /cgi-bin hits point to legacy vectors. Inspect responses and block offenders.

## Conclusion

Traversal/CGI probing listed; update deny rules and signatures.

# 6. Webshell Upload and Access Indicator

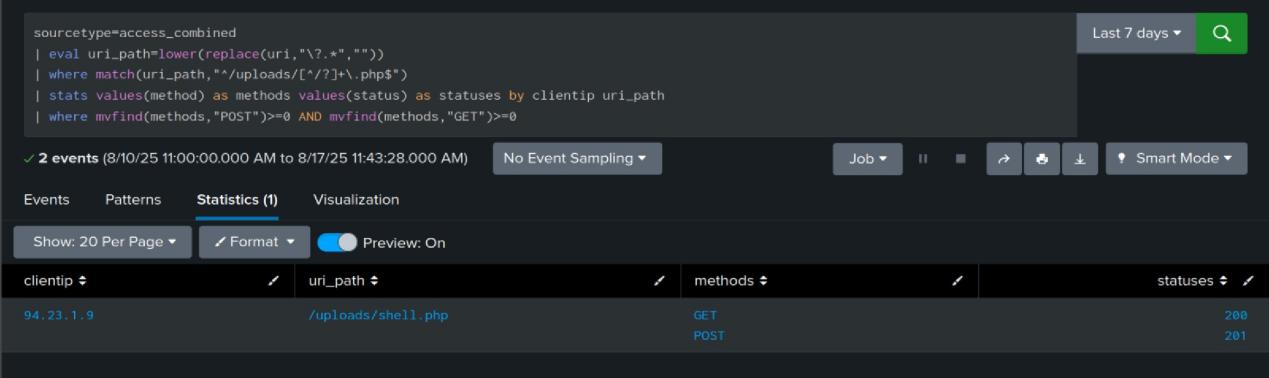
## Objective

Flag potential webshell upload followed by GET access under /uploads/.

## Splunk Query

sourcetype=access\_combined | eval uri\_path=lower(replace(uri,"\?.\*","")) | where match(uri\_path,"^/uploads/[^/?]+\.php$") | stats values(method) as methods values(status) as statuses by clientip uri\_path | where mvfind(methods,"POST")>=0 AND mvfind(methods,"GET")>=0

## Result



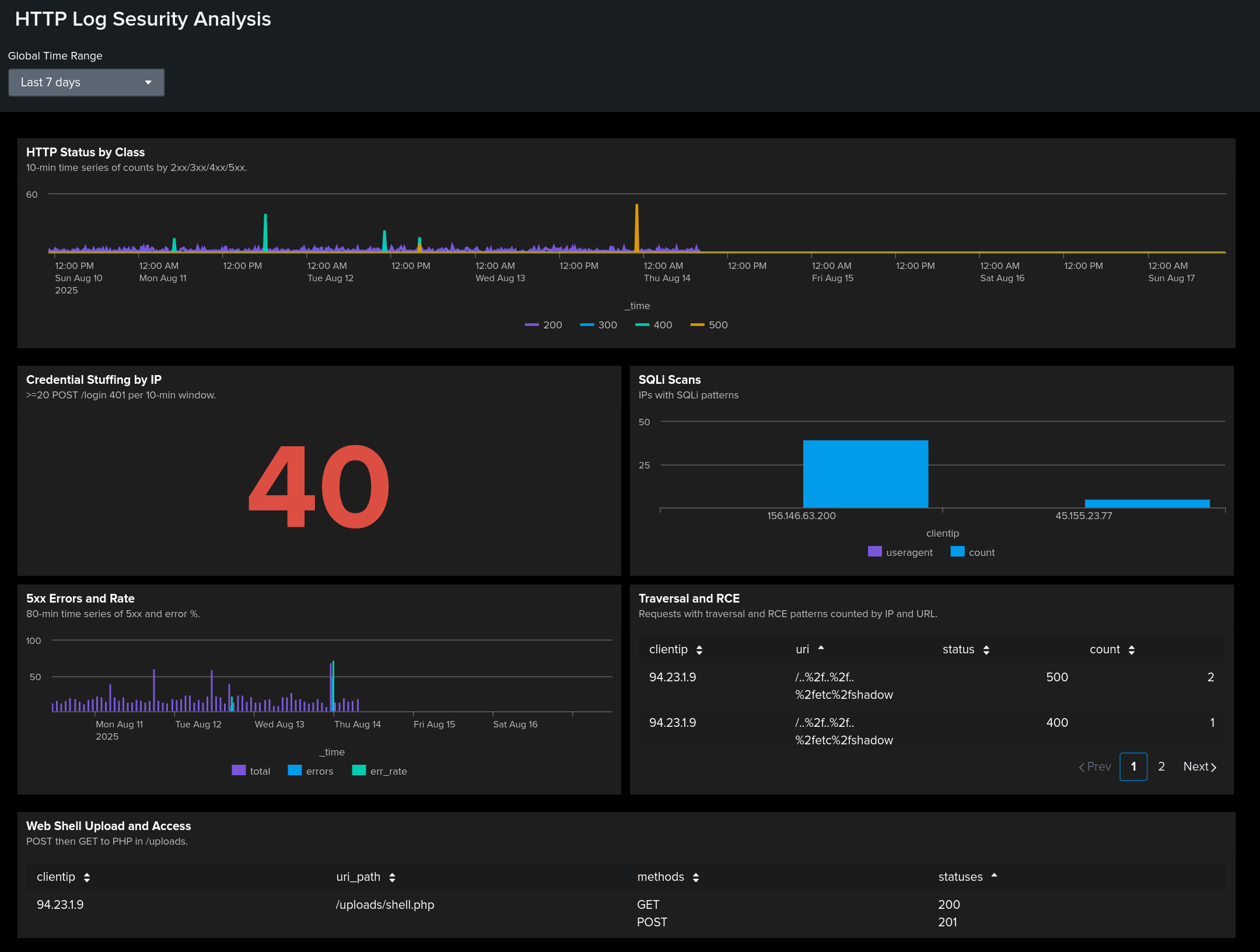
## Analysis

Combined POST+GET on a PHP under /uploads suggests successful upload and execution. Urgent triage required.

## Conclusion

Suspected webshell paths isolated; contain, acquire forensics, rotate secrets.

**Dashboard**



**Dashboard: HTTP Log Security Analysis**

**Scope:** last 7 days. Data: index=main sourcetype=access\_combined.

**Panels:**

**HTTP Status by Class** - 10-min 2xx/3xx/4xx/5xx trend

**Credential Stuffing** - IPs with ≥20 POST /login 401s per 10-min window.

**SQLI Scans** - sqlmap UAs and SQLi payload patterns by IP.

**5xx Errors & Rate** - total vs 5xx and error %.

**Traversal/RCE** - ../, encoded traversal, /cgi-bin/, ?cmd= hits by IP and URL.

**Web Shell Upload/Access** - POST then GET to PHP in /uploads/.

**Triggers**

≥20 failed POST /login per IP/10 min - investigate/block.

Any hit in SQLI or Traversal/RCE - tighten WAF/rate-limit.

Web-shell signal - isolate host, collect artifact, rotate creds.

**Outcome:** Fast triage of attack sources, compromise windows, and risky file ops.

# Global Findings

Traffic has clear baselines with distinct periods of elevated 5xx errors, concentrated brute-force, and automated scanning against injection and traversal vectors.

# Recommendations

• Set SLOs for error rate; alert on deviations.

• Enforce rate limits and lockouts on /login; require MFA.

• Block known scanner UAs; tighten WAF SQLi rules.

• Block traversal patterns and /cgi-bin; monitor cmd parameters.

• Disable execution in upload directories; quarantine suspicious files; rotate secrets.

# Appendix: Full SPL Listing

1. sourcetype="access\_combined" | eval status\_class=floor(status/100)\*100 | timechart span=10m count by status\_class

2. sourcetype="access\_combined" | timechart span=80m count as total count(eval(status>=500)) as errors | eval err\_rate=round(100\*errors/total,2)

3. index=main sourcetype=access\_combined method=POST uri="/login" status=401 | bin \_time span=10m | stats count as fails by clientip \_time | where fails>=20 | sort - fails

4. sourcetype=access\_combined | search useragent=sqlmap\* OR uri="\*' OR 1=1\*" OR uri="\*UNION\*SELECT\*" | stats count by clientip useragent | sort - count

5. sourcetype=access\_combined | search uri="\*../\*" OR uri="\*%2f..\*" OR uri="\*/cgi-bin/\*" OR uri="\*?cmd=\*" | stats count by clientip uri status | sort - count

6. sourcetype=access\_combined | eval uri\_path=lower(replace(uri,"\?.\*","")) | where match(uri\_path,"^/uploads/[^/?]+\.php$") | stats values(method) as methods values(status) as statuses by clientip uri\_path | where mvfind(methods,"POST")>=0 AND mvfind(methods,"GET")>=0