

OWASP Top 10 Attack Detection and Risk Assessment through Splunk Log Analysis

Date: September 16, 2025, 10:00 AM IST

Status: Open – Investigation

1. Executive Summary:

A 24-hour Splunk log analysis of authentication logs (auth.log), web server logs (apache_access.log), and application logs (webapp.json) revealed multiple security events mapped to OWASP Top 10 risks and MITRE ATT&CK techniques.

- i. Authentication logs showed SSH brute-force attempts and suspicious logins, suggesting possible credential compromise.
- ii. Web server logs indicated SQL injection, XSS, and path traversal probes, consistent with automated exploitation tools (e.g., sqlmap).
- iii. Application logs confirmed repeated unauthorized API access attempts and 500 server errors, which may expose backend details.

Overall Risk: **HIGH** – While no confirmed breach was found, repeated exploitation attempts and suspicious logins significantly raise compromise probability.

2. Scope of Analysis:

1. Time Range Analyzed: Last 24 hours (2025-09-15 00:00:00 to 2025-09-16 00:00:00 UTC)
2. Index Used: practice
3. Source Types Analyzed:
 - linux_secure → auth.log
 - access_combined → apache_access.log
 - _json → webapp.json

3. Findings:

1.Authentication Logs (auth.log):

1) Brute-Force Attack:

Severity: **High**

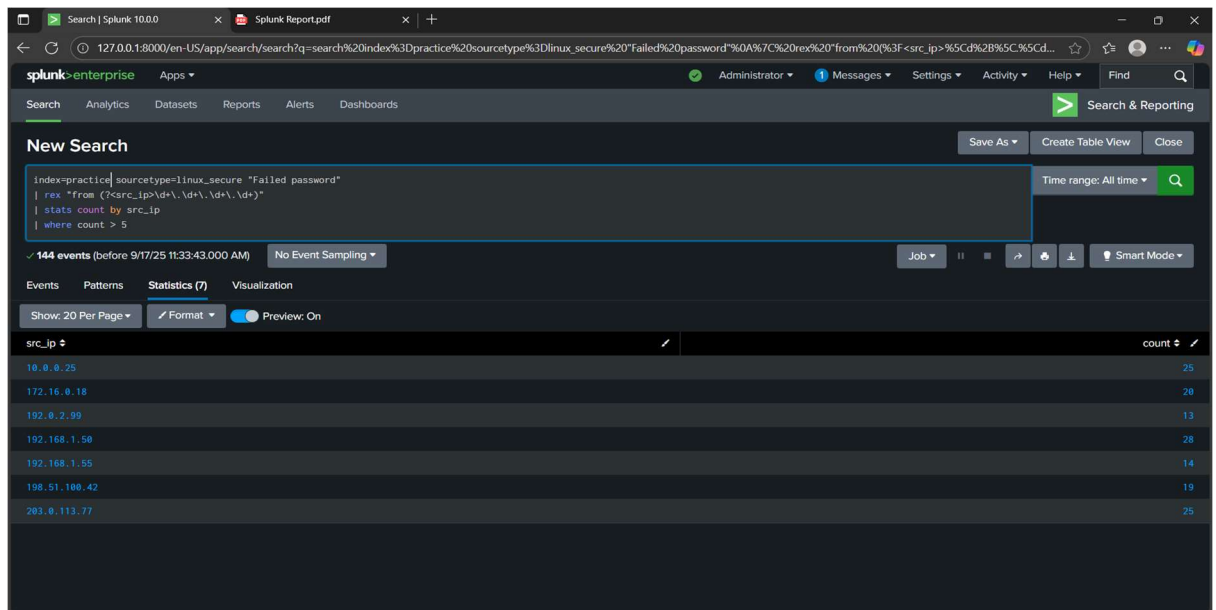
SPL Query:

```
index="practice" sourcetype="linux_secure" "failed password"

| rex "from (?<src_ip>\d+\.\d+\.\d+\.\d+)"

| stats count by src_ip

| where count > 5
```



Analysis:

- Attackers systematically try many passwords against one or a few accounts to discover valid credentials. This is noisy and often automated.
- In Linux auth logs you'll see repeated Failed password entries for the same src_ip and user in a short window. If successful, it may progress to a shell or other access method.
- Brute force is a core credential-access technique (ATT&CK T1110).

Observed:

- What to expect in results: rows of src_ip with high count (failed attempts). Example interpretation: src_ip=198.51.100.23 count=50 → repeated failures from that source.
- Tuning: adjust where count > 5 by timeframe (e.g., earliest=-15m latest=now) to reduce false positives from legitimate users mistyping passwords.

Mapping:

- MITRE ATT&CK: T1110 (Brute Force).
- OWASP Top 10: A07 (Identification & Authentication Failures — permits brute-force).

Risk:

- Compromise of user accounts (especially if password reuse occurs). Potential lateral movement, privilege escalation, data exfiltration.

Action:**1. Preventive:**

- Enforce MFA for all interactive logins and high-privilege accounts.
- Implement strong password policy and block commonly used passwords.
- Rate-limit authentication attempts and configure progressive backoff / lockout policies.

2. Detection / tuning:

- Use time-bounded searches (e.g., earliest=-15m) and baseline expected failed attempt rates.
- Correlate failures with target account importance (alert higher for admin accounts).

3. Response playbook:

- Immediate: block offending IP at perimeter (if external), add to temporary denylist, force password reset for hit accounts if success suspected.
- Enrich IP with threat intel, check for other hits (VPN, web app logs).
- If success observed, begin full account/host compromise triage.

4. Longer term:

- Require MFA, monitor for lateral authentication, audit logs for post-compromise activity.

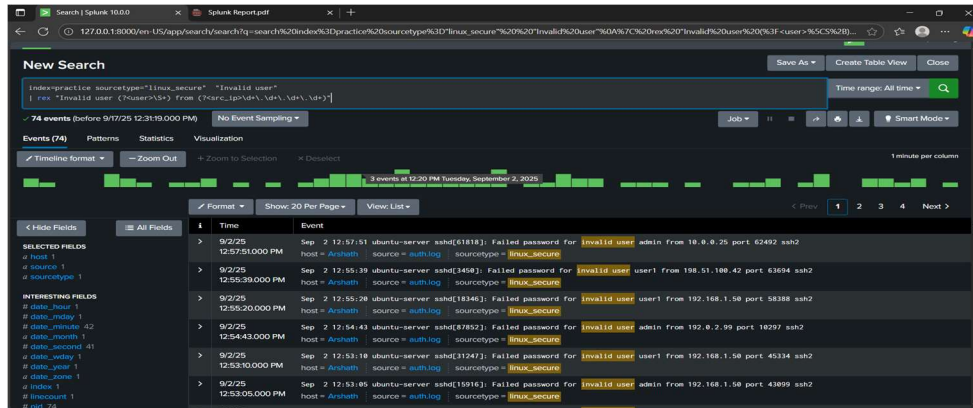
2) Username Enumeration:

Severity: Medium → High

SPL Query:

```
index=practice sourcetype="linux_secure" "Invalid user"
```

```
| rex "Invalid user (?<user>\S+) from (?<src_ip>\d+\.\d+\.\d+\.\d+)"
```



Analysis:

- Enumeration finds valid usernames; attackers use this to build a set of targets for credential spraying/credential stuffing.
- Linux Invalid user messages indicate login attempts where the target username does not exist (or is rejected) — that pattern itself allows attackers to test existence differences (valid versus invalid responses).
- Enumeration is part of account discovery tactics (ATT&CK account discovery / user enumeration).

Observed:

- Expect output rows of user and src_ip showing which usernames are being probed. Example: user=admin src_ip=203.0.113.44.
- Watch for many distinct user values from a single src_ip → indicates systematic enumeration.

Mapping:

- MITRE ATT&CK: Account Discovery / User Enumeration (see T1087 / related account discovery techniques).
- OWASP Top 10: A07 — enumeration supports credential stuffing and other auth attacks.

Risk:

- Attackers obtain lists of valid account names — increases success of later authentication attacks (credential stuffing, targeted phishing).

Action:**1.Preventive:**

- Avoid leaking user existence differences: unify authentication error messages (don't reveal "invalid user" vs "bad password").
- Implement rate limits and bot detection on login endpoints.

2. Detection:

- Alert on src_ip or user agents that probe many distinct accounts in a short period.
- Correlate with OSINT / threat intel to see if usernames correspond to privileged accounts.

3. Response:

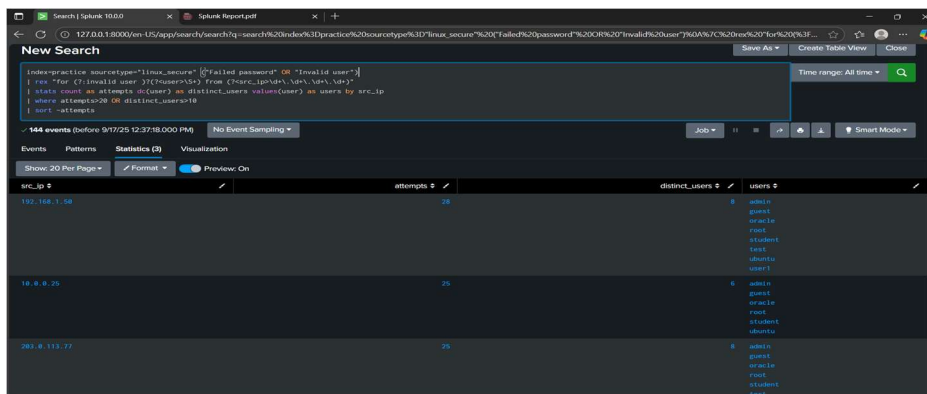
- Block or challenge traffic sources performing mass enumeration (CAPTCHA, WAF rules).
- Monitor enumerated accounts for suspicious subsequent auth attempts.

3) Credential Stuffing:

Severity: **High**

SPL Query:

```
index="practice" sourcetype="linux_secure" ("failed password" OR "Invalid user")  
| rex "for(?:invalid user)?(?:<user>\S+) from(?:<src_ip>\d+\.\d+\.\d+\.\d+)"  
| stats count as attempts dc(user) as distinct_users values(user) as user by src_ip  
| where attempts>20 OR distinct_users>10  
| sort -attempts
```



Analysis:

- Attackers use breached username/password pairs against your service (often with automation). Unlike brute force, credential stuffing uses known passwords from other breaches and targets many accounts quickly.
- Indicators: many failed password or Invalid user events but with large dc(user) (distinct usernames) and many attempts per source IP; or many successes after using credential lists.
- MITRE classifies credential stuffing as a sub-technique of Brute Force (T1110.004). OWASP explicitly calls out credential stuffing as an automated attack permitted by weak authentication controls.

Observed:

- Expected output: per src_ip stats showing attempts and distinct_users. Example interpretation: src_ip=198.51.100.12 attempts=350 distinct_users=240 → strong indicator of stuffing using credential lists.
- Consider adding | where attempts>threshold AND distinct_users>threshold and timeframe filters to reduce noise.

Mapping:

- MITRE ATT&CK: T1110.004 — Credential Stuffing.
- OWASP Top 10: A07 (Identification & Authentication Failures — cites credential stuffing).

Risk:

- High chance of account takeover, especially for users who reuse passwords across services. Successful takeover can lead to data theft, business fraud, or pivoting into internal networks.

Action:**1. Preventive:**

- Compulsory MFA on high-value accounts and ideally for all users.
- Monitor and block known breached credentials (use breached password lists).
- Implement device fingerprinting and progressive friction (CAPTCHA/challenges).

2. Detection:

- Correlate attempts against blacklisted credential pairs or known breach sources.
- Alert when distinct_users and attempts exceed normal baselines.

3. Response:

- Block offending IPs / rate limit; invalidate sessions if success detected; force password resets for affected accounts.
- Notify users if their account was targeted or accessed; recommend credential changes and MFA enrollment.

4. Playbook addition:

- Use automated enrichment of the credential lists used (check hash matches against known breaches).

4) Successful Unauthorized Access:

Severity: Critical

SPL Query:

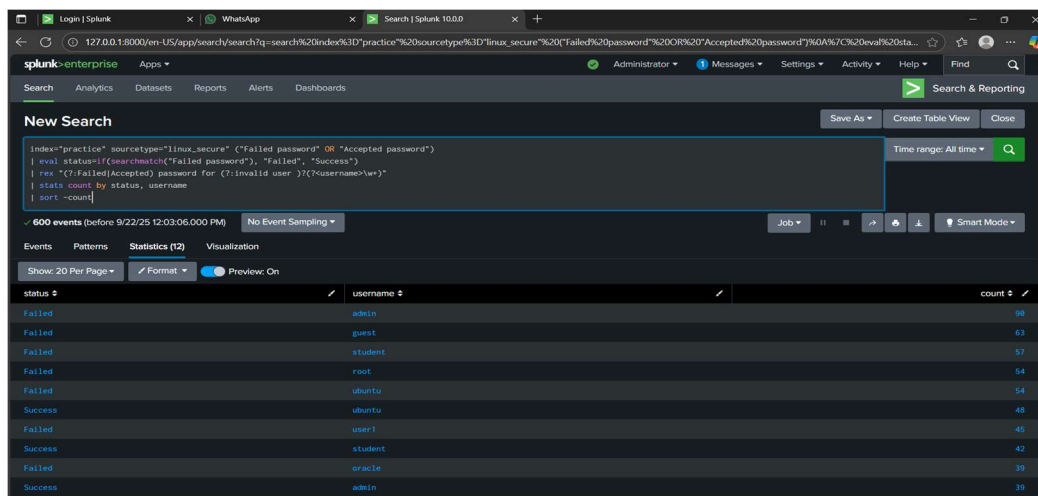
```
index="practice" sourcetype="linux_secure" ("Failed password" OR "Accepted password")
```

```
| eval status=if(searchmatch("Failed password"), "Failed", "Success")
```

```
| rex "(?:Failed|Accepted) password for (?:invalid user )?(?<username>\w+)"
```

```
| stats count by status, username
```

| sort -count



The screenshot shows the Splunk Enterprise web interface. A search query is entered in the 'New Search' bar: `index="practice" sourcetype="linux_secure" ("Failed password" OR "Accepted password") | eval status=if(searchmatch("Failed password"), "Failed", "Success") | rex "(:Failed|Accepted) password for (:invalid user)?(:?username)?" | stats count by status, username | sort -count`. The search results are displayed in a table with columns 'status' and 'username'. The table shows counts for various usernames, with 'admin' having the highest count (98).

status	username	count
Failed	admin	98
Failed	guest	63
Failed	student	57
Failed	root	54
Failed	ubuntu	54
Success	ubuntu	48
Failed	user1	45
Success	student	42
Failed	oracle	39
Success	admin	39

Analysis:

- A sudden increase in Success counts for a user (especially after many failures) may indicate account takeover.
- Successful logins from unexpected geolocations, at odd hours, or from IPs with previous brute/credential attempts are high risk.

Observed:

- Look for status=Success rows for usernames with prior failures or from suspect src_ip. Add fields like src_ip, host, session_id, and geoip to make decisions.
- Example: username=jdoe status=Success count=3 combined with previous Failed attempts from same src_ip → escalate.

Mapping:

- MITRE ATT&CK: Successful use of Valid Accounts (T1078) and T1110 outcomes (if gained via brute force).
- OWASP Top 10: A07 (Identification & Authentication Failures).

Risk:

- Confirmed account compromise → immediate business impact (sensitive data exposure, privilege escalation, lateral movement).

Action:

1. Immediate:

- If login is unauthorized: revoke active sessions, force password reset, disable account if high-risk.
- Collect session metadata (IP, user agent, device fingerprint) and preserve logs for forensic analysis.

2. Investigation:

- Map timeline: prior failures, MFA failure/success, other host activity.
- Look for post-login activity (file access, admin actions, outbound connections).

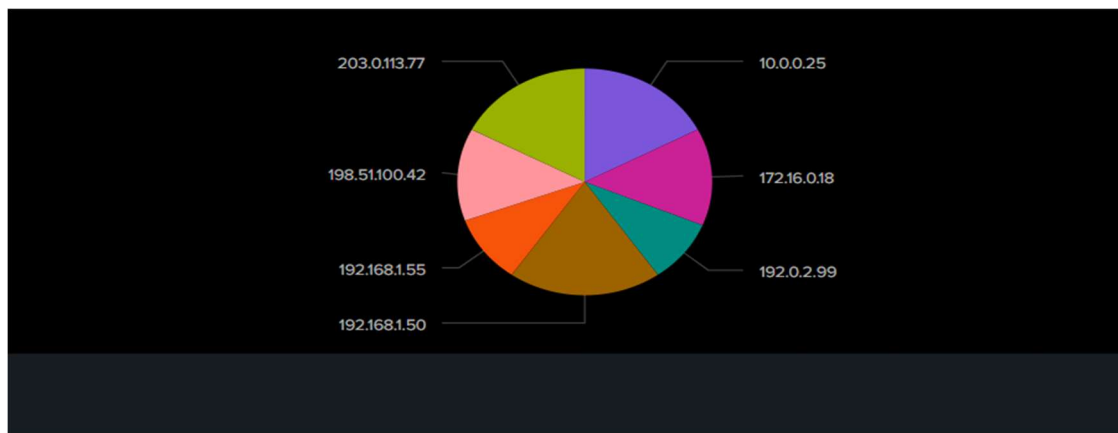
3. Remediation:

- Reset credentials, require MFA, review and rotate keys/tokens. If lateral movement suspected, isolate affected hosts and escalate to IR team.

Dashboard Analysis:

The Splunk dashboards provide a visual representation of the queries executed during the log analysis. Each panel highlights key events and patterns observed across the collected logs, enabling faster detection and easier interpretation of security-related activities.

I. Brute-Force Attack



II. Username Enumeration



IP Range	Percentage
10.0.0.25	10.0%
203.0.113.77	20.3%
192.168.1.50	19.2%
192.0.2.99	19.8%
198.51.100.42	19.5%
172.16.0.18	17.2%
192.168.1.55	19.2%

status	count
Failed	95
Failed	65
Failed	58
Failed	55
Failed	55
Success	48
Failed	45
Success	42
Failed	40
Success	40
Success	40
Failed	30

1) Injection (SQLi / classic injection):

SPL Query:

```
index=practice sourcetype=access_combined
| rex field=_raw "\"(?<method>GET|POST) (?<uri_full>[^\s]+) HTTP/"
| eval uri_query = coalesce(uri_query, uri_full)
| where match(uri_query,"(?i)UNION\\s+SELECT")
    OR match(uri_query,"(?i)information_schema")
    OR match(uri_query,"(?i)concat\\(")
    OR match(uri_query,"(?i)sleep\\(")
    OR match(uri_query,"(?i)benchmark\\(")
    OR match(uri_query,"(?i)'\\s*or\\s*'1'='1'")
    OR match(useragent,"(?i)sqlmap")
| stats count by clientip, uri_query, useragent, _time
| sort -count
```

```

index=practice sourcetype=access_combined
| rex field=_raw "(?<method=GET|POST) (?<uri_full>[^\s]+) HTTP/"
| eval uri_query = coalesce(uri_query, uri_full)
| where match(uri_query, "(?i)UNION\\s+SELECT")
      OR match(uri_query, "(?i)information_schema")
      OR match(uri_query, "(?i)concat\\(")
      OR match(uri_query, "(?i)sleep\\(")
      OR match(uri_query, "(?i)benchmark\\(")
      OR match(uri_query, "(?i)\\s+or\\s+'1'='1'")
      OR match(useragent, "(?i)sqlmap")
| stats count by clientip, uri_query, useragent, _time
| sort -count

```

✓ 39 events (before 9/22/25 2:31:46.000 PM) No Event Sampling ▼
Job ▾ || ▮ → ⚙ ⬇ ⚡ Fast Mode ▾

Events Patterns **Statistics (37)** Visualization

Show: 20 Per Page ▾
Format ▾ ☒ Preview: On

 < Prev 1 2 Next >

clientip	uri_query	useragent	_time	count
192.168.1.18	name<script>alert(1)</script>	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	2
192.168.1.24	name<script>alert(1)</script>	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	2
192.168.1.10	/index.html	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	1
192.168.1.11	/index.html	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	1
192.168.1.13	id=1	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	1
192.168.1.15	q=UNION+SELECT+user,password+FROM+users	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	1
192.168.1.16	id=1	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	1
192.168.1.17	/dashboard	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	1
192.168.1.19	name<script>alert(1)</script>	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	1
192.168.1.21	/login	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	1
192.168.1.21	q=UNION+SELECT+user,password+FROM+users	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	1
192.168.1.22	id=1	sqlmap/1.5.28dev (http://sqlmap.org)	2025-09-08 20:38:31	1

Analysis:

Our SPL looks for typical SQL injection indicators (UNION SELECT, information_schema, concat(), sleep(), benchmark(), ' or '1'='1, sqlmap user-agents). These are reliable indicators of active exploitation attempts or automated scanning/probing for SQL injection vulnerabilities on public-facing web apps. Presence of sleep()/benchmark() indicates time-based injection attempts (blind SQLi). UNION SELECT and information_schema indicate attempts to enumerate or exfiltrate database schema/data.

Observed:

- Fields: clientip, uri_query, useragent, _time (from your stats).
- Example observations to expect: requests containing UNION SELECT, information_schema, concat(, sleep(5), or user-agent strings with sqlmap. Multiple hits from same IP or different URIs suggest scanning; hits with legitimate cookies/session identifiers may indicate targeted exploitation.

Mapping:

- I. MITRE ATT&CK Mapping: Exploit Public-Facing Application (technique families that cover web app exploitation and injection-based data access). Also maps to Data from Information Repositories and Exfiltration techniques when successful.
- II. OWASP Top 10: Injection (the Injection category).

Risk:

- High risk of data disclosure (database contents), authentication bypass, data manipulation, or full compromise of the application backend. Successful SQLi can lead to data breach, privilege escalation, and persistent backdoors.

Action:

- **Detection / Triage:**

- Create an alert when count of matching events from a single IP > X within Y minutes (suggest default: 5 events / 5 minutes).
- Correlate with successful HTTP 200 / 302 responses and large response sizes to spot data exfiltration.
- Enrich with reputation lookup for clientip, geolocation, and known proxy/VPN lists.
- **Immediate Response:** block offending IPs at the firewall or WAF if repeated and obviously malicious; disable any accounts that show unusual database activity.
- **Remediation:** perform input validation and parameterized queries (prepared statements), apply least privilege to DB accounts, fix vulnerable endpoints. Deploy/verify WAF rules covering UNION SELECT and common payload encodings.
- **Hunt & Post-mortem:** check DB logs for suspicious queries, review backups for data integrity, rotate credentials if exfiltration suspected.
- **Tune Splunk:** decode URL-encodings (you already use coalesce) and add regex for common evasion encodings (e.g., %20, comments, case variations).

2) Cross-Site Scripting (XSS):

Severity: Medium → High (depends on context: reflected XSS is often medium; stored XSS is high)

SPL Query:

```
index=practice sourcetype=access_combined
| rex field=_raw "\"(?:<method>GET|POST) (?:<uri_full>[^\s]+) HTTP/"
| eval uri_decoded = urldecode(coalesce(uri_query, uri_full))
| eval uri_to_check = lower(coalesce(uri_decoded, uri_full))
| where match(uri_to_check,"<script>")
  OR match(uri_to_check,"alert\\(\"")
  OR match(uri_to_check,"onerror=")
  OR match(uri_to_check,"javascript:")
| stats count by clientip, uri_full, referer, _time
| sort -count
```

<pre> index=practice sourcetype=access_combined rex field=_raw "(?<method>GET POST) (?<uri_full>[^\s]+) HTTP/" eval uri_decoded = urldecode(coalesce(uri_query, uri_full)) eval uri_to_check = lower(coalesce(uri_decoded, uri_full)) where match(uri_to_check, "<script>") OR match(uri_to_check, "alert\\(") OR match(uri_to_check, "onerror=") OR match(uri_to_check, "javascript:") stats count by clientip, uri_full, referer, _time sort -count </pre>					Time range: All time
✓ 18 events (before 9/22/25 2:31:46.000 PM) No Event Sampling					Job
Events Patterns Statistics (15) Visualization					
Show: 20 Per Page Format Preview: On					
clientip	uri_full	referer	_time	count	
192.168.1.18	/page?name=<script>alert(1)</script>	http://example.com	2025-09-08 20:38:31	2	
192.168.1.28	/page?name=<script>alert(1)</script>	http://evil.com	2025-09-08 20:38:31	2	
192.168.1.32	/page?name=<script>alert(1)</script>	http://google.com	2025-09-08 20:38:31	2	
192.168.1.10	/page?name=<script>alert(1)</script>	http://evil.com	2025-09-08 20:38:31	1	
192.168.1.14	/page?name=<script>alert(1)</script>	http://evil.com	2025-09-08 20:38:31	1	
192.168.1.18	/page?name=<script>alert(1)</script>	http://evil.com	2025-09-08 20:38:31	1	
192.168.1.18	/page?name=<script>alert(1)</script>	http://google.com	2025-09-08 20:38:31	1	
192.168.1.19	/page?name=<script>alert(1)</script>	http://evil.com	2025-09-08 20:38:31	1	

Analysis:

Our SPL decodes URIs and searches for <script>, alert(, onerror=, javascript: — good for detecting reflected/script insertion attempts. Findings indicate injection of client-side script into URLs or parameters; if the app reflects those inputs unsanitized back to other users, stored XSS risk exists.

Observed:

- Fields: clientip, uri_full, referer, _time.
- Expect requests with URL-encoded scripts (e.g., %3Cscript%3E), or onerror=alert(1) in images/parameters, or javascript: pseudo-protocol. Look for repeated attempts, variations (capitalization, encoded payloads), and suspicious referers.

Mapping:

- I. MITRE ATT&CK Mapping: Exploitation of Client Execution / Road to Credential Theft via client-side script execution; can also enable Credential Harvesting and Session Token theft.
- II. OWASP Top 10: Cross-Site Scripting (XSS).

Risk:

- Reflected XSS: can lead to targeted phishing and session theft (medium risk).
- Stored XSS: persistent execution against other users (high risk — can lead to site-wide compromise or mass credential theft).

Action:

- **Detection / Triage:** alert on repeated script-bearing requests, especially to parameters that render into HTML templates. Flag if referer is internal (indicates potential reflected content).

- **Immediate Response:** if exploit appears stored, take the vulnerable page offline or disable user inputs until fixed. Remove malicious stored content from DB.
- **Remediation:** implement proper output encoding/escaping, Content Security Policy (CSP), input sanitization, and use safe template engines.
- **Prevention:** use frameworks' anti-XSS features, CSP with script-nonce, and sanitize HTML input if needed.
- **Tune Splunk:** add detection for encoded variants and event normalization (lowercase + urldecode — you already do this).

3) Path traversal / Local File Inclusion (LFI/RFI):

Severity: **High**

SPL Query:

```
index=practice sourcetype=access_combined
```

```
| eval uri_full = coalesce(uri_query, uri_path)
```

```
| search uri_full="*../*" OR uri_full="*..%2F*" OR uri_full="*/etc/passwd*" OR uri_full="*%2Fetc%2Fpasswd*"
uri_full="*%2Fetc%2Fpasswd"
```

```
| stats count by clientip, uri_full, _time
```

```
| sort -count
```

The screenshot shows a Splunk search interface with the following SPL query:

```
index=practice sourcetype=access_combined
| eval uri_full = coalesce(uri_query, uri_path)
| search uri_full="*../*" OR uri_full="*..%2F*" OR uri_full="*/etc/passwd*" OR uri_full="*%2Fetc%2Fpasswd*"
| stats count by clientip, uri_full, _time
| sort -count
```

The results table shows 17 events. The columns are clientip, uri_full, _time, and count. The data indicates multiple attempts from various IP addresses to access files like ../../etc/passwd and ../../etc/passwd%2Fetc%2Fpasswd.

clientip	uri_full	_time	count
192.168.1.36	../../etc/passwd	2025-09-08 20:38:31	4
192.168.1.21	../../etc/passwd	2025-09-08 20:38:31	2
192.168.1.26	../../etc/passwd	2025-09-08 20:38:31	2
192.168.1.10	../../etc/passwd	2025-09-08 20:38:31	1
192.168.1.13	../../etc/passwd	2025-09-08 20:38:31	1
192.168.1.14	../../etc/passwd	2025-09-08 20:38:31	1
192.168.1.15	../../etc/passwd	2025-09-08 20:38:31	1
192.168.1.17	../../etc/passwd	2025-09-08 20:38:31	1
192.168.1.23	../../etc/passwd	2025-09-08 20:38:31	1
192.168.1.24	../../etc/passwd	2025-09-08 20:38:31	1
192.168.1.28	../../etc/passwd	2025-09-08 20:38:31	1

Analysis:

Our SPL searches for ../, encoded ../%2F, and /etc/passwd references — classic path traversal and local file read attempts. Presence of etc/passwd patterns is a direct attempt to enumerate or read local files. RFI attacks could attempt to include remote payloads.

Observed:

- Fields: clientip, uri_full, _time.
- Indicators: requests containing sequences like ../../.., %2e%2e%2f, or explicit /etc/passwd. Look for extension chaining (e.g., .php?file=../../..../etc/passwd) and user agents that are scanners.

Mapping:

- I. MITRE ATT&CK Mapping: Exploit Public-Facing Application (file inclusion/abuse leading to code execution or data discovery); could map to Local File Discovery and Collection techniques.
- II. OWASP Top 10: Broken Access Control / Security Misconfiguration (depending on root cause) and also relates to Injection/LFI categories in OWASP community guidance.

Risk:

- High: successful traversal/LFI can expose sensitive files (passwords, keys), lead to code execution (if inclusion leads to code eval), or allow full system compromise.

Action:

- Detection / Triage: alert on any hits to */etc/passwd* and repeated ../ attempts. If combined with suspicious response codes (200 with small content), pull the response body to check if file content was returned.
- Immediate Response: block source IPs showing multiple LFI attempts; check the server for presence of dropped web shells; isolate affected hosts if suspicious.
- Remediation: restrict file access (canonicalize and sanitize path inputs), use safe APIs (deny .. in file paths), set proper filesystem permissions, disable dangerous include features or remote file inclusion, ensure PHP allow_url_include=Off.
- Hunt: search webserver logs for successful reads of sensitive files and check system logs for privilege escalation signs.

4) Sensitive Data Exposure (secrets in URLs / responses):

Severity: High (if secrets found in URLs/responses)

SQL Query:

index=practice sourcetype=access_combined

| search uri_query="*password*" OR uri_query="*token*" OR uri_query="*Authorization*" OR uri_query="*passwd*"

| stats count by _time, clientip, method, uri_query, status, useragent, referer

index=practice sourcetype=access_combined search uri_query="*password*" OR uri_query="*token*" OR uri_query="*Authorization*" OR uri_query="*passwd*" stats count by _time, clientip, method, uri_query, status, useragent, referer							
Time range: All time							
✓ 19 events (before 9/22/25 2:31:46.000 PM) No Event Sampling							
Events Patterns Statistics (19) Visualization							
Show: 20 Per Page Format Preview: On							
_time	clientip	method	uri_query	status	useragent	referer	count
2025-09-08 20:38:31	192.168.1.11	POST	q=UNION+SELECT+user,password+FROM+users	301	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)	http://example.com	1
2025-09-08 20:38:31	192.168.1.11	POST	q=UNION+SELECT+user,password+FROM+users	500	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)	http://google.com	1
2025-09-08 20:38:31	192.168.1.13	GET	q=UNION+SELECT+user,password+FROM+users	301	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)	http://google.com	1
2025-09-08 20:38:31	192.168.1.13	GET	q=UNION+SELECT+user,password+FROM+users	401	curl/7.68.0	http://evil.com	1
2025-09-08 20:38:31	192.168.1.15	GET	q=UNION+SELECT+user,password+FROM+users	301	sqlmap/1.5.2#dev (http://sqlmap.org)	http://evil.com	1
2025-09-08 20:38:31	192.168.1.15	POST	q=UNION+SELECT+user,password+FROM+users	200	Mozilla/5.0 (X11; Linux x86_64)	http://evil.com	1
2025-09-08 20:38:31	192.168.1.17	GET	q=UNION+SELECT+user,password+FROM+users	500	Mozilla/5.0 (X11; Linux x86_64)	http://evil.com	1
2025-09-08 20:38:31	192.168.1.17	POST	q=UNION+SELECT+user,password+FROM+users	200	Mozilla/5.0 (Windows NT 10.0; Win64; x64)	-	1
2025-09-08 20:38:31	192.168.1.21	POST	q=UNION+SELECT+user,password+FROM+users	401	sqlmap/1.5.2#dev (http://sqlmap.org)	http://google.com	1
2025-09-08 20:38:31	192.168.1.28	POST	q=UNION+SELECT+user,password+FROM+users	200	Mozilla/5.0 (X11; Linux x86_64)	http://evil.com	1

Analysis:

Our SPL searches URIs for password, token, Authorization, passwd — this catches sensitive values sent in query strings or visible in logged responses. Presence of secrets in URLs is a common developer mistake (tokens in GET parameters) and causes data leakage via logs, referer headers, browser history, and proxies.

Observed:

- Fields: _time, clientip, method, uri_query, status, useragent, referer.
- Observations: querystrings like ?password=..., ?token=eyJ..., Authorization= in URI (rare but sometimes misconfigured). Also check for 200 responses that echo back secrets.

Mapping:

- I. MITRE ATT&CK Mapping: Credential Access (disclosure of secrets); Data from Information Repositories if secrets leak to logs or external systems.
- II. OWASP Top 10: Sensitive Data Exposure.

Risk:

- High: leaked secrets (API tokens, passwords) may allow account takeover, API misuse, lateral movement, and data exfiltration.

Action:

- Detection / Triage: alert on any URL containing password, token, Authorization, etc. Immediately identify which endpoint/service is leaking.
- Immediate Response: invalidate leaked tokens/rotate credentials for affected services, force password reset for impacted accounts if necessary.

- Remediation: stop sending secrets in URLs — move to POST bodies or use secure cookies/authorization headers; do not log sensitive parameters (mask or drop them in web server and application logs). Implement TLS everywhere.
- Preventive Controls: token expiry, least privilege, central secrets management, and scanning codebase for instances where secrets might be constructed into URLs.
- Splunk tuning: add field-extraction to mask sensitive values and create a scheduled audit to identify any future occurrences.

5) Security Misconfiguration (scanners / known tools):

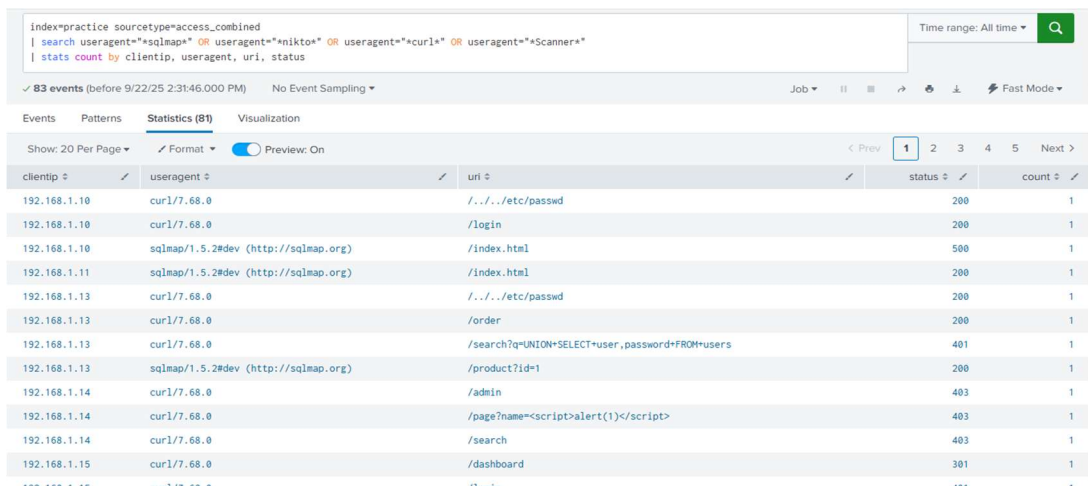
Severity: **Medium**

SQL Query:

index=practice sourcetype=access_combined

```
| search useragent="*sqlmap*" OR useragent="*nikto*" OR useragent="*curl*" OR useragent="*Scanner*"
```

```
| stats count by clientip, useragent, uri, status
```



The screenshot shows a Splunk search interface with the following query: `index=practice sourcetype=access_combined | search useragent="*sqlmap*" OR useragent="*nikto*" OR useragent="*curl*" OR useragent="*Scanner*" | stats count by clientip, useragent, uri, status`. The results are displayed in a table with columns: clientip, useragent, uri, status, and count. The table shows 83 events.

clientip	useragent	uri	status	count
192.168.1.10	curl/7.68.0	/../../../../etc/passwd	200	1
192.168.1.10	curl/7.68.0	/login	200	1
192.168.1.10	sqlmap/1.5.2#dev (http://sqlmap.org)	/index.html	500	1
192.168.1.11	sqlmap/1.5.2#dev (http://sqlmap.org)	/index.html	200	1
192.168.1.13	curl/7.68.0	/../../../../etc/passwd	200	1
192.168.1.13	curl/7.68.0	/order	200	1
192.168.1.13	curl/7.68.0	/search?q=UNION+SELECT+user,password+FROM+users	401	1
192.168.1.13	sqlmap/1.5.2#dev (http://sqlmap.org)	/product?id=1	200	1
192.168.1.14	curl/7.68.0	/admin	403	1
192.168.1.14	curl/7.68.0	/page?name=<script>alert(1)</script>	403	1
192.168.1.14	curl/7.68.0	/search	403	1
192.168.1.15	curl/7.68.0	/dashboard	301	1
192.168.1.15	curl/7.68.0	/login	401	1

Analysis:

Our SPL searches user-agent strings for known scanner tools (sqlmap, nikto, curl, Scanner). These hits usually indicate reconnaissance or automated scanning activity. Alone they are not successful exploitations, but they are a good early-warning indicator of probing and often precede targeted attacks.

Observed:

- Fields: clientip, useragent, uri, status.
- Look for sqlmap, nikto, curl, Scanner in useragent. Observe request patterns (many different URIs in short time) and response codes (403/404 vs 200) to assess whether scanning succeeded.

Mapping:

- I. MITRE ATT&CK Mapping: Reconnaissance / Active Scanning techniques used during initial discovery and vulnerability scanning.
- II. OWASP Top 10: Security Misconfiguration (identifies scanners and misconfigured endpoints that leak info); also related to "A6:2021 Vulnerable and Outdated Components" in some contexts.

Risk:

- Medium: indicates an attempt to discover vulnerabilities; successful scans can reveal entry points. Repeated scanning may be precursor to an exploit.

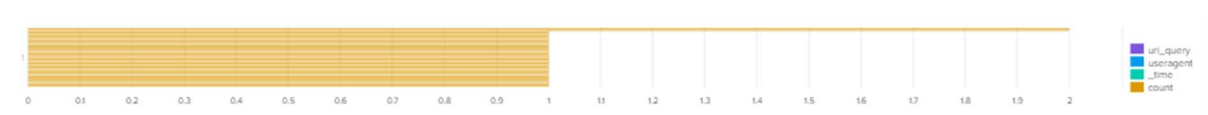
Action:

- Detection / Triage: alert on bursty useragent values containing known scanner names, particularly when combined with high request rates or requests to admin endpoints.
- Immediate Response: block repeat offenders with rate-limiting or firewall/WAF rules; consider serving a honeypot/redirect for scanners to gather intel.
- Remediation / Prevention: ensure no directory listing, disable default/administrative endpoints, remove debugging endpoints from production, apply up-to-date software and patches, secure server banners.
- Harden: deploy WAF with rules to detect known scanner signatures; configure rate limits; ensure proper error handling (do not leak stack traces).

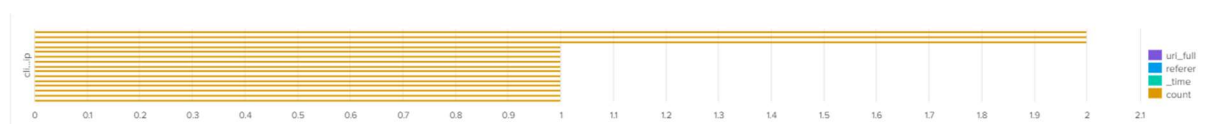
Dashboard Analysis:

The Splunk dashboards provide a visual representation of the OWASP Top 10 attack queries executed during the log analysis. Each panel highlights key events, attacker behaviors, and suspicious payloads detected in the apache_access.log.

I. Injection (SQLi / classic injection) (A03 / A1)



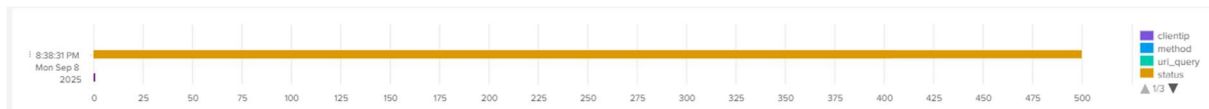
II. Cross-Site Scripting (XSS) (A07/A4)



III. Path traversal / Local File Inclusion (LFI/RFI) (A05 / A3)



IV. Sensitive Data Exposure (secrets in URLs / responses) (A02)



V. Security Misconfiguration (scanners / known tools) (A05)



3. Application Log (webapp.json):

1) Unhandled Exceptions (500):

Severity: **High**

SPL Query:

index=practice sourcetype=_json

| spath status

| spath endpoint

| search status=500

| stats count BY endpoint

<pre>index=practice sourcetype=_json spath status spath endpoint search status=500 stats count BY endpoint</pre>		Time range: All time
✓ 70 events (before 9/17/25 7:45:44.000 AM) No Event Sampling		
Events Patterns Statistics (6) Visualization		
Show: 20 Per Page Format Preview: On		
endpoint	count	
/api/admin	6	
/api/cart	6	
/api/checkout	8	
/api/login	14	
/api/order	16	
/api/search	20	

Analysis:

The query identified HTTP 500 errors across application endpoints. Results show frequent unhandled exceptions, with /api/search having the highest failures, followed by /api/login and /api/checkout. These errors indicate backend issues that may affect system reliability, user access, and transaction processing.

Observed:

- High occurrence of 500 errors was observed across multiple endpoints.
- The /api/search endpoint recorded the maximum unhandled exceptions, indicating potential instability or input validation issues.
- Errors in /api/login could prevent users from accessing the application, affecting availability.
- Failures in /api/checkout may directly disrupt transaction flow, leading to financial and operational risks.
- Consistent 500 responses indicate a lack of error handling and may require deeper investigation into application code, API dependencies, or database interactions.

Mapping:

- I. MITRE ATT&CK Mapping: T1190 — Exploit Public-Facing Application
- II. OWASP Top 10: A6 — Security Misconfiguration, A9 — Using Components with Known Vulnerabilities

Risk:

- HTTP 500 errors may reveal backend exceptions or stack traces, which attackers can leverage for reconnaissance.
- Repeated failures in critical endpoints like /api/search, /api/login, and /api/checkout may lead to denial of service, credential access issues, or business disruption.

Action:

- Mask error responses: Return generic error messages to the client, log details internally.
- Fix application logic: Patch code handling for /api/search, /api/login, /api/checkout to prevent unhandled exceptions.
- Apply monitoring & alerting: Track spikes in 500 errors as potential signs of attack or system instability.
- Regular updates: Patch libraries, dependencies, and frameworks to reduce the chance of exploitable errors.

2) Unauthorized / Forbidden (401/403):

Severity: Medium

SPL Query:

index=practice sourcetype=_json

| spath status

| spath endpoint

| search status=401 OR status=403

| stats count BY user, endpoint

The screenshot shows a Splunk search interface with the title 'Unauthorized / Forbidden (401/403)'. The search bar contains the query: `index=practice sourcetype=_json | spath status | spath endpoint | search status=401 OR status=403 | stats count BY user, endpoint`. Below the search bar, there are tabs for 'Events', 'Patterns', 'Statistics (25)', and 'Visualization'. The 'Statistics' tab is selected, showing a table with columns 'user' and 'endpoint'. The table lists various users and endpoints with their corresponding counts. For example, 'alice' has 8 counts for '/api/login', 'bob' has 10 counts for '/api/login', and 'charlie' has 12 counts for '/api/login'.

user	endpoint	count
alice	/api/login	8
alice	/api/cart	4
alice	/api/checkout	4
alice	/api/login	4
alice	/api/order	4
alice	/api/search	6
anonymous	/api/login	10
anonymous	/api/cart	14
anonymous	/api/checkout	4
anonymous	/api/login	8
anonymous	/api/order	10
anonymous	/api/search	6
bob	/api/login	10
bob	/api/cart	10
bob	/api/checkout	8
bob	/api/login	4
bob	/api/order	2
bob	/api/search	8
charlie	/api/login	12
charlie	/api/cart	8

Analysis:

The query analyzed application logs (webapp.json) to identify unauthorized (401) and forbidden (403) requests. The results show multiple failed access attempts across different endpoints and users. Frequent 401s suggest failed authentication attempts, while 403s indicate users trying to access resources without proper permissions.

Observation:

- Several users generated repeated 401 Unauthorized errors, indicating possible invalid login attempts.
- 403 Forbidden responses were seen on sensitive endpoints, suggesting attempts to access restricted resources.
- High volume of such errors could indicate brute-force login attempts or privilege escalation attempts.

Mapping:

I. MITRE ATT&CK Mapping:

- T1078 — Valid Accounts (attackers attempting to use or guess valid credentials).

- T1110 — Brute Force (repeated failed logins).

II. OWASP Top 10:

- A2 — Broken Authentication
- A5 — Broken Access Control

Risk:

- Repeated unauthorized requests may indicate credential stuffing or brute-force attacks.
- Unauthorized access attempts to restricted APIs may lead to data exposure or privilege escalation if misconfigurations exist.
- If not monitored, attackers may eventually gain valid credentials and compromise accounts.

Action:

- Implement account lockout / rate limiting after repeated failed login attempts.
- Strengthen authentication with MFA (Multi-Factor Authentication).
- Restrict access: Enforce role-based access control (RBAC) for sensitive endpoints.
- Monitor & alert on spikes in 401/403 errors as potential brute-force indicators.
- Review audit logs for suspicious user activity and failed login patterns.

3) Attack Attempts (SQLi, XSS, Path Traversal):

Severity: High

SPL Query:

```
index=practice sourcetype=_json ("SQL Injection" OR "XSS" OR "Path traversal")
```

```
| eval attack_type=case(
```

```
    like(message,"%SQL Injection%"), "SQL Injection",
```

```
    like(message,"%XSS%"), "Cross-Site Scripting",
```

```
    like(message,"%Path traversal%"), "Path Traversal",
```

```
    true(), "Other"
```

)

| stats count values(user) as Users values(endpoint) as Endpoints by attack_type

| sort -count

```
index=practico sourcetype=son ("SQL Injection" OR "XSS" OR "Path Traversal")
| eval attack_type=case(
  like(message,"SQL Injection"), "SQL Injection",
  like(message,"XSS"), "Cross-Site Scripting",
  like(message,"Path Traversal"), "Path Traversal",
  true(), "Other"
)
| stats count values(user) as Users values(endpoint) as Endpoints by attack_type
| sort -count
```

attack_type	count	Users	Endpoints
Path Traversal	88	alice anonymous bob charlie guest	/api/admin /api/cart /api/checkout /api/login /api/order /api/search
SQL Injection	72	alice anonymous bob charlie guest	/api/admin /api/cart /api/checkout /api/login /api/order /api/search
Cross-Site Scripting	58	alice anonymous bob charlie guest	/api/admin /api/cart /api/checkout /api/login /api/order /api/search

Analysis:

The query identified malicious payload attempts in the application logs, specifically targeting SQL Injection (SQLi), Cross-Site Scripting (XSS), and Path Traversal vulnerabilities. Multiple endpoints and users were flagged as sources of such attacks. This indicates active probing of the application for exploitable weaknesses.

Observation:

- Path Traversal attempts were observed most frequently, suggesting attackers are trying to access sensitive directories or files.
- SQL Injection payloads indicate attempts to manipulate backend databases via vulnerable parameters.
- XSS attempts were recorded, showing attempts to inject malicious scripts into web responses.
- These repeated attacks highlight that the application is actively being targeted by external threat actors.

Mapping:

I. MITRE ATT&CK Mapping:

- T1190 — Exploit Public-Facing Application
- T1505 — Server Software Component

II. OWASP Top 10:

- A1 — Injection (SQL Injection)
- A3 — Sensitive Data Exposure / Injection-based access
- A5 — Broken Access Control
- A7 — Cross-Site Scripting (XSS)

Risk:

- SQL Injection could lead to unauthorized data access, modification, or database takeover.
- XSS could allow attackers to steal session cookies, credentials, or perform malicious redirections.
- Path Traversal may expose sensitive server files (e.g., /etc/passwd), enabling privilege escalation.
- If successful, these attacks can result in data breaches, account compromise, or full system takeover.

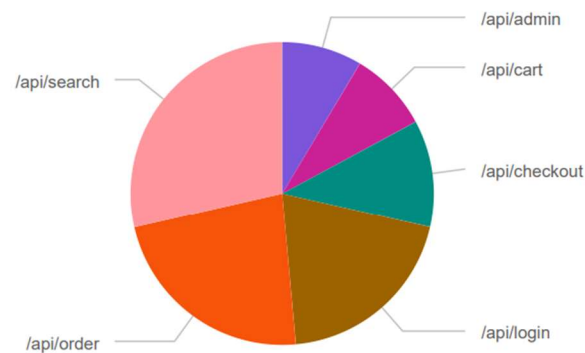
Action:

- Input validation & sanitization: Enforce strict server-side validation on all user inputs.
- Use parameterized queries / ORM to prevent SQL Injection.
- Implement WAF (Web Application Firewall) rules to detect and block common attack payloads.
- Apply output encoding to mitigate XSS.
- Restrict file access permissions and validate file paths to prevent traversal attacks.
- Continuous monitoring & alerting for abnormal request patterns.

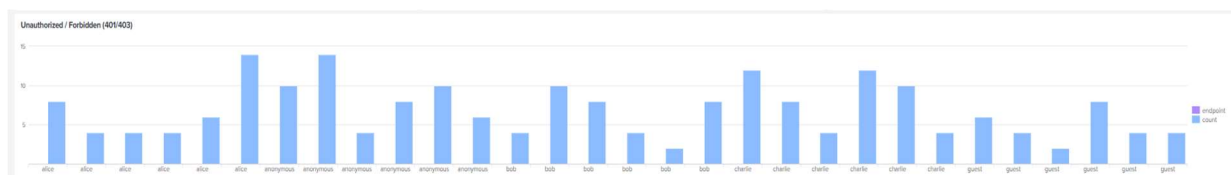
Dashboard Analysis:

The Splunk dashboards provide a visual representation of the queries executed during the log analysis. Each panel highlights key events and patterns observed across the collected logs, enabling faster detection and easier interpretation of security-related activities.

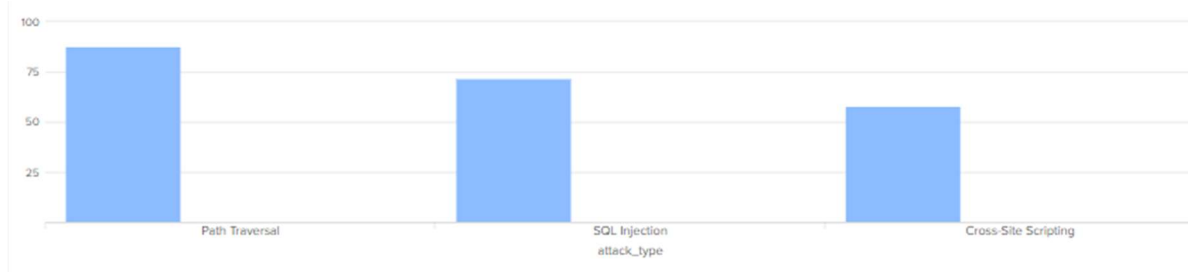
I. Unhandled Exceptions (500)



II. Unauthorized / Forbidden (401/403)



III. Attack Attempts (SQLi, XSS, Path Traversal)



4. Appendices

Exported Files (Hypothetical):

- I. Internal Server Errors (500) - [InternalServerError.csv](#)
- II. Unauthorized / Forbidden Access (401/403) - [UnauthorizedForbiddenAccess.csv](#)
- III. Web Attacks (SQLi, XSS, Path Traversal Attempts) - [Webattacks.csv](#)