🔐 Detecting Brute Force Attacks in Linux Using Splunk Correlation Search & Alerts

🖊 **By Faizanullah Syed**
Faizanullah.syed19@gmail.com
SOC Analyst Trainee | Cybersecurity Enthusiast | Founder, UnmaskCyber.com

◈ 1. Objective

The goal of this assignment is to:

- Detect brute force login attempts from system authentication logs using Splunk.
- Use correlation logic to identify attack patterns.
- Create a Splunk alert that triggers when suspicious behaviour (e.g., ≥ 5 failed login attempts from same IP/user) is detected.
- Demonstrate actionable threat monitoring using Splunk.

◈ 2. Environment Used

Component	Details
Splunk Server	Splunk Enterprise (or Free)
Universal Forwarder	Installed on Linux system (Ubuntu/RHEL)
Data Source	`/var/log/auth.log`
Hostname	`splunkufserver`
Splunk Index	`main` or custom
Sourcetypes	`auth-2`, `auth-4`, or file input

◈ 3. What is Correlation in Splunk?

Correlation is the process of connecting different logs or data points to find meaningful patterns that indicate suspicious or malicious behavior.

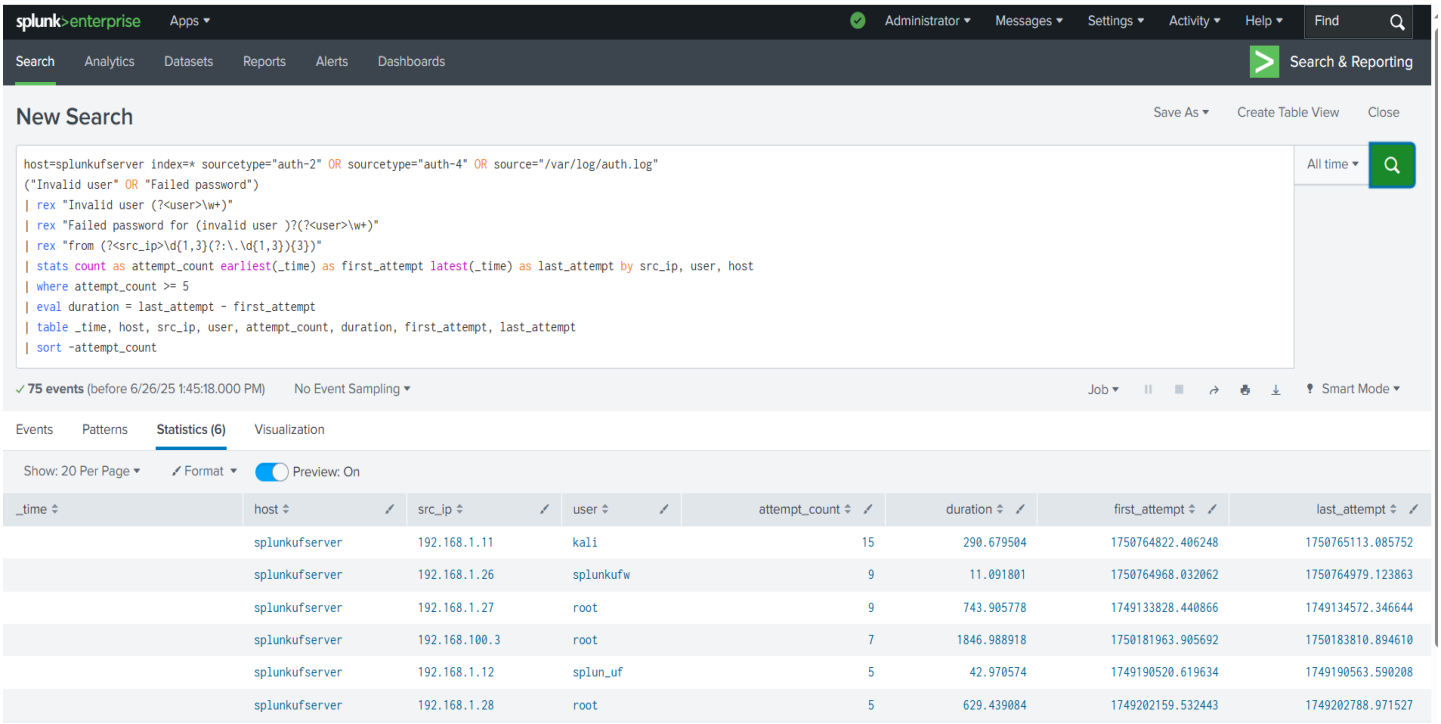
In this case:

- **Multiple failed logins** from the same IP/user within a short time **correlates** to brute-force attempts.
- Detecting that pattern using SPL makes Splunk a powerful tool in SIEM.

4. SPL Query Used (Explained)

Full SPL:

```
host=splunkufserver index=* sourcetype="auth-2" OR sourcetype="auth-4" OR source="/var/log/auth.log"

("Invalid user" OR "Failed password")

| rex "Invalid user (?<user>\w+)"

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

| rex "from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3})"

| stats count as attempt_count earliest(_time) as first_attempt latest(_time) as last_attempt by src_ip, user, host

| where attempt_count >= 5

| eval duration = last_attempt - first_attempt

| table _time, host, src_ip, user, attempt_count, duration, first_attempt, last_attempt

| sort -attempt_count
```



_time	host	src_ip	user	attempt_count	duration	first_attempt	last_attempt
	splunkufserver	192.168.1.11	kali	15	290.679504	1750764822.406248	1750765113.085752
	splunkufserver	192.168.1.26	splunkufw	9	11.091801	1750764968.032062	1750764979.123863
	splunkufserver	192.168.1.27	root	9	743.905778	1749133828.440866	1749134572.346644
	splunkufserver	192.168.100.3	root	7	1846.988918	1750181963.905692	1750183810.894610
	splunkufserver	192.168.1.12	splun_uf	5	42.970574	1749190520.619634	1749190563.590208
	splunkufserver	192.168.1.28	root	5	629.439084	1749202159.532443	1749202788.971527

🔍 Line-by-Line Breakdown:

Line	Explanation
`host=splunkufserver index=* ...`	Filters logs from a specific host and auth log source types.
`"Invalid user" OR "Failed password"`	Matches logs with failed login patterns.
`rex "Invalid user (?<user>\w+)"`	Extracts username from "Invalid user" lines.
`rex "Failed password for ...`	Extracts username from "Failed password" lines.
`rex "from (?<src_ip>...)`	Extracts source IP of the login attempt.
`stats count ...`	Groups results by IP and user; counts failed attempts.
`where attempt_count >= 5`	Filters only those users/IPs with 5+ failed attempts.
`eval duration = ...`	Calculates how long the attack lasted.
`table ...`	Creates clean tabular output.
`sort -attempt_count`	Sorts highest attempt count first.

◈ 5. Sample Output Table

Events Patterns **Statistics (6)** Visualization

Show: 20 Per Page ✓ Format ✓ Preview: On

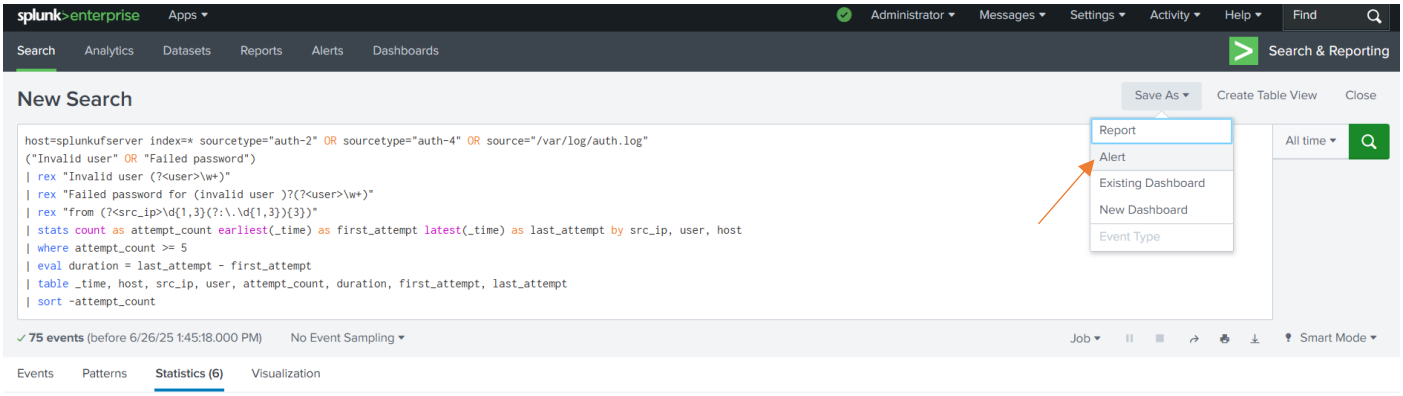
_time	host	src_ip	user	attempt_count	duration	first_attempt	last_attempt
	splunkufserver	192.168.1.11	kali	15	290.679504	1750764822.406248	1750765113.0857
	splunkufserver	192.168.1.26	splunkufw	9	11.091801	1750764968.032062	1750764979.1238
	splunkufserver	192.168.1.27	root	9	743.905778	1749133828.440866	1749134572.3466
	splunkufserver	192.168.100.3	root	7	1846.988918	1750181963.905692	1750183810.8946
	splunkufserver	192.168.1.12	splun_uf	5	42.970574	1749190520.619634	1749190563.5902
	splunkufserver	192.168.1.28	root	5	629.439084	1749202159.532443	1749202788.9715

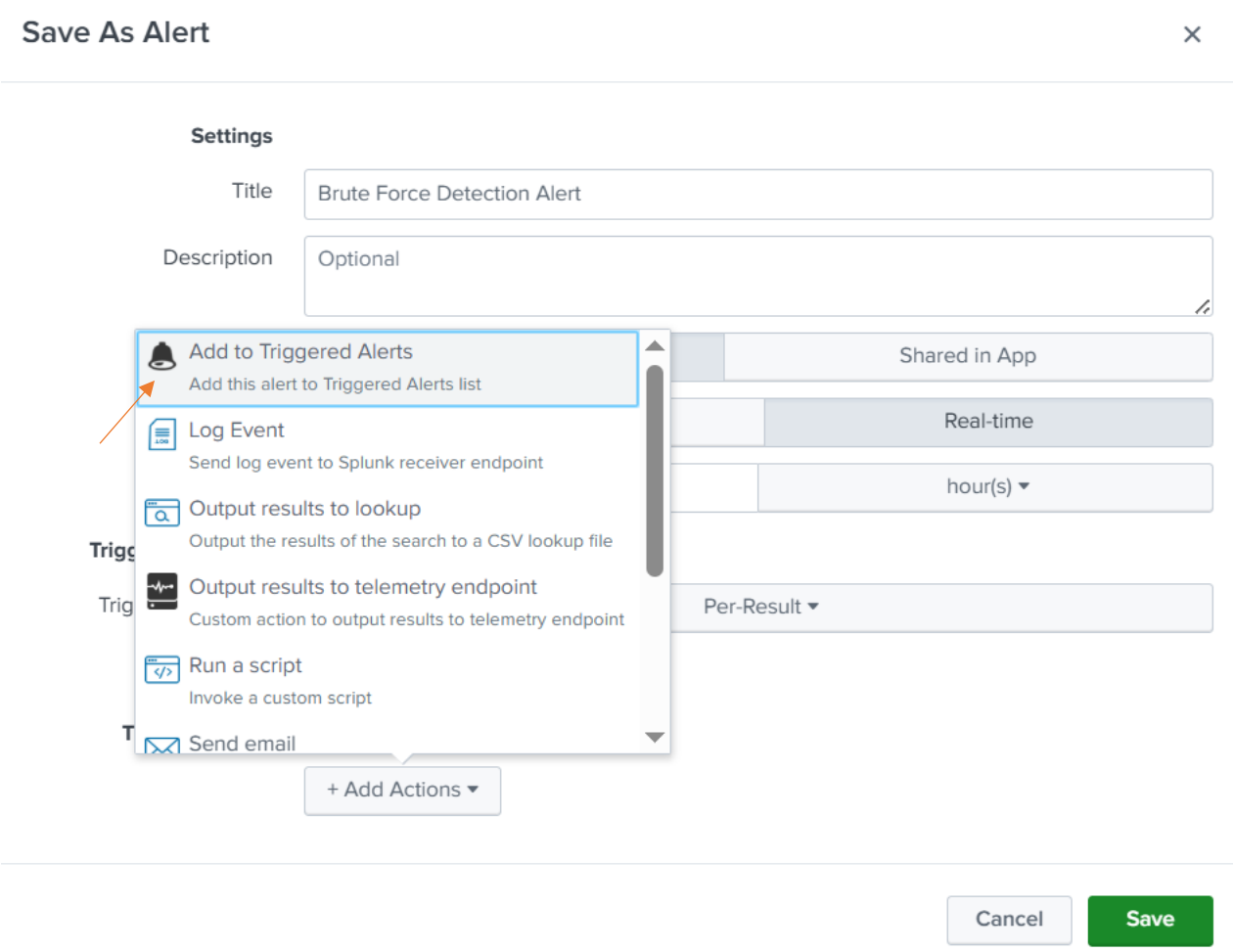
◈ 6. Creating the Alert in Splunk

Steps to Create Alert

1. Run your SPL in **Search & Reporting** app.
2. Click **Save As > Alert**.
3. Fill in the following:

Field	Value
Alert Name	Brute Force Detection Alert
Schedule	Every 5 minutes
Trigger condition	Number of results > 0
Time range	Last 5 minutes
Alert Type	Scheduled alert
Trigger Action	Send email / webhook / create notable event
Severity	High





Save As Alert

Description: Optional

Permissions: Private | Shared in App

Alert type: Scheduled | Real-time

Expires: 24 hour(s)

Trigger Conditions

Trigger alert when: Per-Result

Throttle ?

Trigger Actions

+ Add Actions

When triggered: Add to Triggered Alerts — Remove

Severity: High

Cancel | Save

Edit Alert

Settings

Alert: **Brute Force Detection Alert**

Description: Optional

Output the results of the search to a CSV lookup file

Output results to telemetry endpoint
Custom action to output results to telemetry endpoint

Run a script
Invoke a custom script

Send email
Send an email notification to specified recipients

Send to Splunk Mobile
Send a notification to Splunk Mobile recipients

Webhook
Generic HTTP POST to a specified URL

Real-time

hour(s)

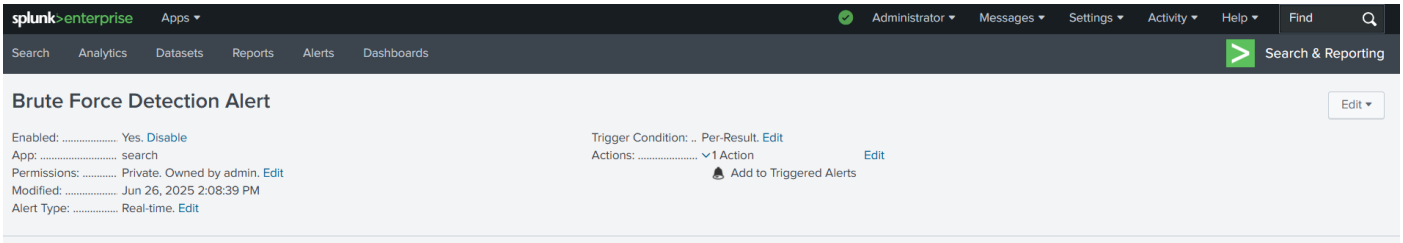
Per-Result

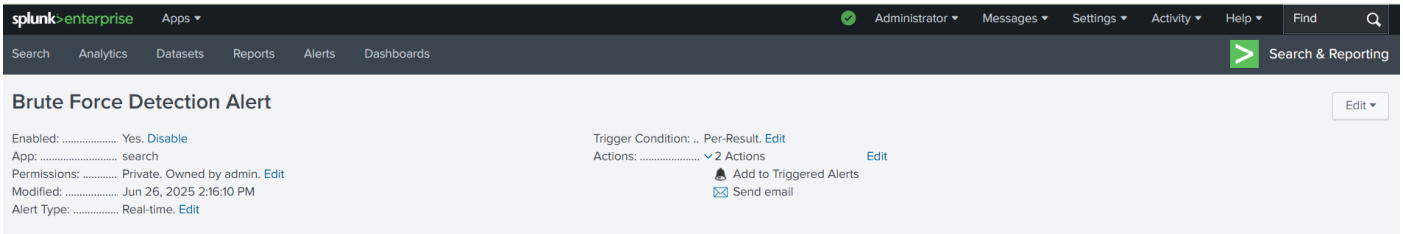
+ Add Actions

When triggered: Add to Triggered Alerts — Remove

Severity: High

Cancel | Save





4. Example email:
   - **Subject**: 🚨 Brute Force Attempt Detected from $result.src_ip$
   - **Body**: Alert triggered on $result.host$ for user `$result.user$` with `$result.attempt_count$` attempts.

🔷 7. Real-World Relevance

✅ Use Case:

This alert helps identify early stages of brute-force attacks, where an attacker tries different usernames and passwords repeatedly to gain access to a Linux system.

📉 Without Alerting:

- The attack might succeed, and you'd only know after compromise.

🛡️ With Correlation + Alert:

- You get a real-time alert.
- Can auto-block IP by using a SOAR action or firewall.
- Respond quickly before damage is done

🔷 8. Conclusion

This project shows how Splunk can be used as a real-time threat detection system. Using simple log data and smart correlation, you can detect brute-force attacks and take proactive action. This forms a key SOC Analyst responsibility.