Sumo Logic dashboards — Geo Location of Console Logins & CrowdStrike Threat Intelligence (organized)

Overview

This document captures your step-by-step process (cleaned and ordered) for building two dashboards in Sumo Logic:

1. **Geo Location of Console Logins** — visualizes console login geolocation, failed logins, brute-force detection and impossible travel (land-speed) detections.
2. **CrowdStrike / Threat Intelligence** — checks IPs against a threat feed (sumo://threat/cs) and highlights outliers.

   Prerequisites: working Sumo Logic account, CloudTrail logs indexed under `_sourceCategory=Labs/AWS/CloudTrail`, geo lookup available (`geo://location`), and a threat index `sumo://threat/cs` with fields (`type`, `threat`, `raw`, `threatlevel` or similar).

Use conventions

- `{{Event_type}}` and `{{Actor}}` are dashboard template variables (Logs Search type) used to filter panels.
- Each panel section shows the exact Sumo query to paste into the panel query editor.

Dashboard A — Geo Location of Console Logins

1) Create a new dashboard & add the first (time series) panel

1. Dashboard → **Create New Dashboard**.
2. **Add Panel** → choose **Time Series**.
3. Paste this query into the panel query box and **Run**:

```
_sourceCategory=Labs/AWS/CloudTrail
| json field=_raw "userIdentity.userName" as actor
| json field=_raw "eventType" as event_type
| json field=_raw "sourceIPAddress" as src_ip
| json field=_raw "eventName" as event_name
| count by actor, event_type, event_name
| top 30 actor by event_type, _count
| transpose row actor column event_type
```

1. Click **Add to Dashboard** (don’t worry about final panel name yet).

2) Create template (dashboard) variables

Open the dashboard settings: **Show variables** → **Add a new variable**.

Variable 1 — Event_type

- Variable type: **Logs Search**
- Name: `Event_type`
- Query (paste exactly):

```
_sourceCategory="Labs/AWS/CloudTrail"
| json field=_raw "eventType" as event_type
| count by event_type
| fields - _count
```

- Key: `event_type`
- Save the variable.

Variable 2 — Actor

- Variable type: **Logs Search**
- Name: `Actor`
- Query (paste exactly):

```
_sourceCategory=Labs/AWS/CloudTrail
| json field=_raw "userIdentity.sessionContext.sessionIssuer.userName" as
actor
| count by actor
| fields - _count
```

- Key: `actor`
- Save the variable.

  Tip: If you want a broader actor selection, you can provide alternate queries using `userIdentity.userName` instead of `sessionIssuer.userName`.

3) Edit the first panel to use variables

1. **Edit Dashboard** → open the panel → **Edit query**.
2. Replace the panel query with the variable-aware version and **Update** the panel:

```
_sourceCategory=Labs/AWS/CloudTrail
| json field=_raw "userIdentity.userName" as actor
| json field=_raw "eventType" as event_type
| json field=_raw "sourceIPAddress" as src_ip
| json field=_raw "eventName" as event_name
| where event_type matches "{{Event_type}}"
| where actor matches "{{Actor}}"
| count by actor, event_type, event_name
| top 30 actor by event_type, _count
| transpose row actor column event_type
```

This makes the panel respond to the `Event_type` and `Actor` template variables.

---

4) Add a Map panel — Geo Location of Console Logins

1. **Add Panel** → choose **Map**.
2. Use this query and **Run**:

```
_sourceCategory="Labs/AWS/CloudTrail"
| json field=_raw "userIdentity.userName" as actor
| json field=_raw "eventType" as event_type
| json field=_raw "sourceIPAddress" as src_ip
| json field=_raw "responseElements.ConsoleLogin" as result
| where !isEmpty(actor)
| where !isEmpty(event_type)
| where event_type matches "{{Event_type}}"
| where actor matches "{{Actor}}"
| lookup latitude, longitude, country_name, city, region from geo://location
on ip=src_ip
| count by actor, src_ip, latitude, longitude, country_name, event_type
```

1. **Name** this dashboard (if you haven't yet): **Geo Location of Console Logins** and **Add to Dashboard**.

---

5) Panel — Top 10 Number of Failed Login Attempts (table)

1. **Add Panel** → **Time Series** (then change chart type to **Table**).
2. Paste and run this query:

```
_sourceCategory=Labs/AWS/CloudTrail
| json field=_raw "userIdentity.userName" as actor
| json field=_raw "eventType" as event_type
| json field=_raw "sourceIPAddress" as src_ip
| json field=_raw "responseElements.ConsoleLogin" as result
| json field=_raw "eventName" as event_name
| where actor matches "{{Actor}}"
| where !isEmpty(actor)
| where !isEmpty(event_type)
| where result = "Failure"
| where event_type = "AwsConsoleSignIn"
| timeslice 1h
| count by _timeslice, actor, event_type, event_name, result
| top 10 actor by _timeslice, event_type, result,_count
```

1. **Name** the panel: *Top 10 No of Failed Login Attempts* and **Add to Dashboard**.

---

6) Panel — Account Compromise Detection From Brute Force Attack (table)

1. **Add Panel** → **Time Series** (change to **Table**).
2. Use this query and run:

```
_sourceCategory=Labs/AWS/CloudTrail
| json field=_raw "userIdentity.userName" as actor
| json field=_raw "eventType" as event_type
| json field=_raw "sourceIPAddress" as src_ip
| json field=_raw "responseElements.ConsoleLogin" as result
| json field=_raw "eventName" as event_name
| where event_type matches "{{Event_type}}"
| where actor matches "{{Actor}}"
| timeslice 1h
| if(result = "Success",1,0) as success_count
| if(result = "Failure",1,0) as failure_count
| sum(success_count) as Success, sum(failure_count) as Failure by
_timeslice,actor,event_type
| where Failure>0
| Failure/Success*100 as Failure_percent
| order by Failure_percent desc
| format("%.2f", Failure_percent) as Failure_percent
```

1. **Name** the panel: *Account Compromise Detection From Brute Force Attack* and **Add to Dashboard**.

---

7) Panel — Land Speed Violation (impossible travel detection)

1. **Add Panel** → **Time Series** (change to **Table**).
2. Paste and run this advanced query:

```
_sourceCategory=Labs/AWS/CloudTrail
| json "userIdentity.userName","sourceIPAddress" as user, ip nodrop
| where user matches "{{Actor}}"
| where isPublicIP(ip)
| min(_messagetime) as login_time by user, ip
| sort by user, +login_time
| ipv4ToNumber(ip) as ip_decimal
| backshift ip_decimal by user
| backshift login_time as previous_login
| where !(isNull(_backshift))
| toInt(floor(_backshift/pow(256,3))) as octet1
| toInt(floor((_backshift - (octet1 * pow(256,3)))/pow(256,2))) as octet2
| toInt(floor((_backshift - (octet1 * pow(256,3) + octet2 * pow(256,2)))/
256)) as octet3
| toInt(_backshift - (octet1 * pow(256,3) + octet2 * pow(256,2) + octet3 *
256)) as octet4
| concat(octet1, ".", octet2, ".", octet3, ".", octet4) as previous_ip
| lookup latitude as lat1, longitude as long1, country_name as country_name1
from geo://location on ip
| lookup latitude as lat2, longitude as long2, country_name as country_name2
from geo://location on ip=previous_ip
| where !(isNull(lat1)) and !(isNull(long1)) and !(isNull(lat2)) and !
(isNull(long2))
| haversine(lat1, long1, lat2, long2) as distance_kms
| (login_time - previous_login)/3600000 as login_time_delta_hrs
| distance_kms / login_time_delta_hrs as apparent_velocity_kph
| where apparent_velocity_kph > 0
| 500 as suspicious_speed
| where apparent_velocity_kph > suspicious_speed
| concat(ip, ", ", previous_ip) as ip_addresses
| if(country_name1 <> country_name2, concat(country_name1, ",",
country_name2), country_name1) as countries
| fields user, ip_addresses, countries, distance_kms, login_time_delta_hrs,
apparent_velocity_kph
| where !isNull(user)
| where apparent_velocity_kph != "Infinity"
| sort by apparent_velocity_kph
```

1. **Name** the panel: *Land Speed Violation* and **Add to Dashboard**.

   This panel highlights apparent impossible travel (login locations too far apart in too short a time).

Dashboard B — CrowdStrike / Threat Intelligence

8) Create a new dashboard and add CloudStrike/CrowdStrike threat-intel panel

1. Dashboard → **Create New Dashboard** (separate from Geo Location dashboard).
2. **Add Panel** → **Time Series**.
3. Paste this query and **Run**:

```
_sourceCategory=Labs/AWS/CloudTrail
| parse regex "(?<ip_address>\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})" multi
| where ip_address != "0.0.0.0" and ip_address != "127.0.0.1"
| lookup type, actor, raw, threatlevel as malicious_confidence from sumo://
threat/cs on threat=ip_address
| where type="ip_address" and !isNull(malicious_confidence)
| if(isEmpty(actor), "Unassigned", actor) as Actor
| parse field=raw "\"ip_address_types\":[\"*\"]" as ip_address_types nodrop
| parse field=raw "\"kill_chains\":[\"*\"]" as kill_chains nodrop
| timeslice 1m
| count by _timeslice, ip_address, malicious_confidence, Actor, kill_chains,
ip_address_types, _sourceCategory, _source
| fields - ip_address, malicious_confidence, Actor, kill_chains,
ip_address_types, _sourceCategory, _source
| count by _timeslice
| outlier _count window=5, threshold=3, consecutive=1, direction=+-
```

1. **Name** the panel: *Cloud Strike Threat Intelligence* (or *CrowdStrike Threat Intelligence*) and add it to the new dashboard.

9) Export dashboard as PDF

1. From the dashboard view, click **Export** → **PDF** (or use the dashboard menu → Export to PDF).
2. Choose desired options (time range, layout) and export.

Best practices & small improvements

- Use consistent variable keys (`actor`, `event_type`) and consider a default value like `All` to avoid empty filters.
- For high-cardinality `Actor` variables, consider limiting the variable query by `top 200 actor by _count` to avoid a huge template list.
- Add panel descriptions and useful links (runbook, playbook) in panel metadata for analysts.
- Ensure `geo://location` lookup is populated and up-to-date (city/country resolution accuracy affects map and haversine results).
- Validate your threat feed mapping (fields names in `sumo://threat/cs`) — adjust `lookup` join keys if your threat index uses a different field.
- Consider adding alert rules (Monitors) for: large number of failed logins, impossible travel events, and any positive matched threat intelligence hits.
