**🔐 SPLUNK BASICS — SOC LEARNING ROADMAP (LINUX-ONLY) + Labs**

Goal: **Trust data → reconstruct behavior → make decisions**

**🔴 PHASE 0 — DATA TRUST (NON-NEGOTIABLE)**

**Theory**

* Event time vs index time
* What “missing logs” actually means
* Why wrong sourcetype breaks investigations

**Practical**

# 🔴 PHASE 0 PRACTICAL — STEP-BY-STEP (LINUX → SPLUNK)

**Objective :**  
Prove that Linux authentication logs in Splunk are **trustworthy for investigation**.

You are **not hunting attacks**.  
You are validating **reality**.

## 🧪 STEP 1 — PROVE LOGS ARE GENERATED (GROUND TRUTH)

### On the Linux server (terminal)

#### 1️⃣ Check auth log exists

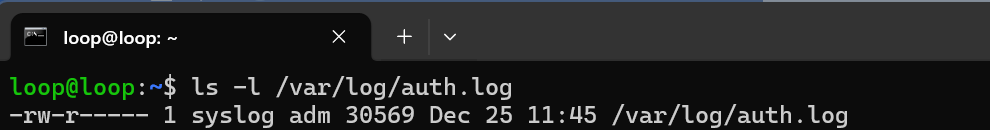
**ls -l /var/log/auth.log**

(or /var/log/secure on RHEL)

You must see:

* File exists
* File size > 0

If not → stop. Logging itself is broken.



#### 2️⃣ Watch logs live

**sudo tail -f /var/log/auth.log**

Leave this running.

#### 3️⃣ Generate known activity (controlled test)

From **another terminal or machine**:

* Attempt SSH login (fail once)
* Then login successfully
* Run a sudo command

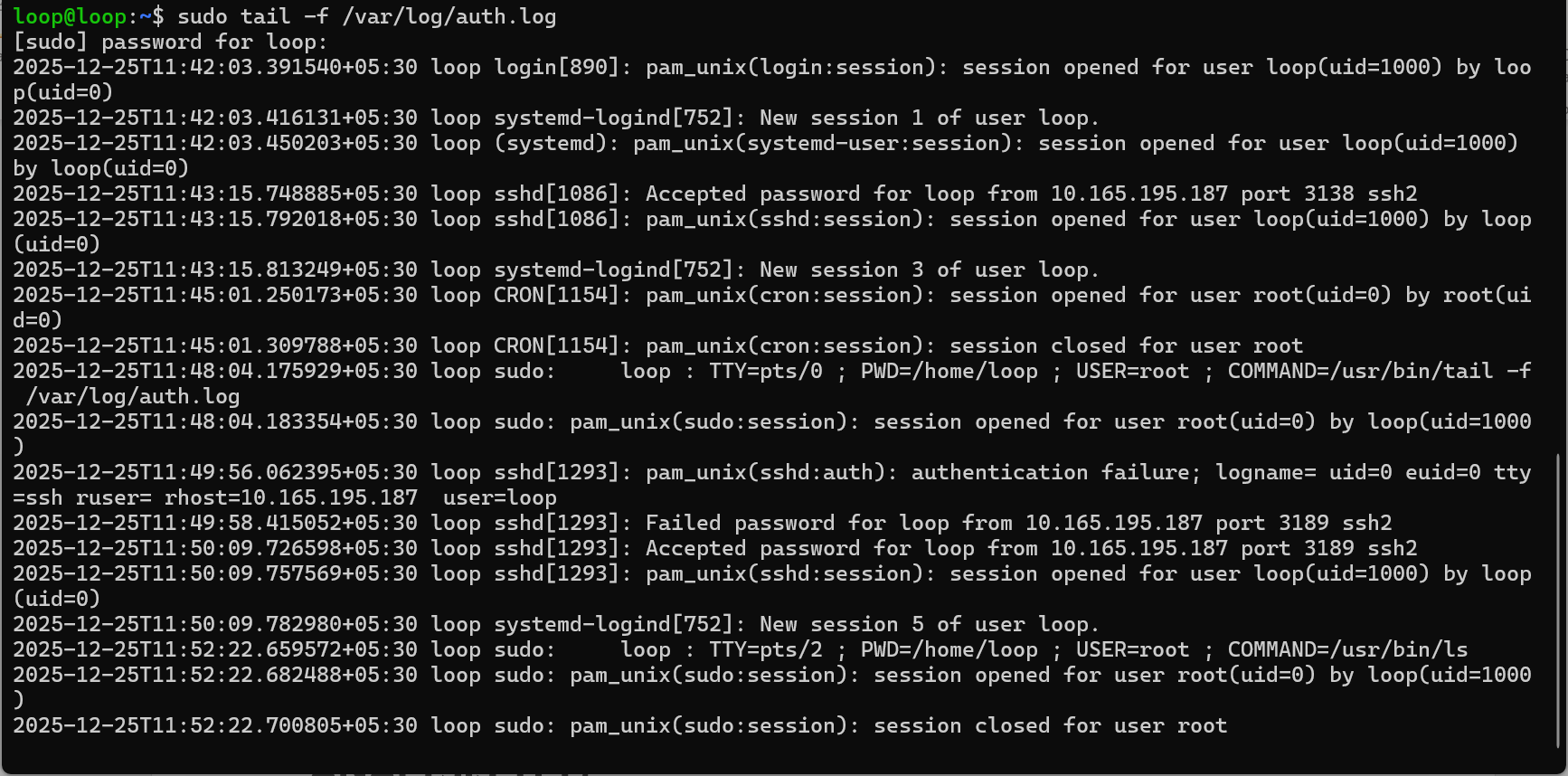
Example:

**ssh wronguser@server\_ip**

**ssh correctuser@server\_ip**

**sudo ls**

👉 Observe new log lines appear **immediately** in auth.log.



### ✅ What you are proving

* Logs are **generated**
* Timestamp looks same
* Format looks consistent

If logs don’t appear here → Splunk is irrelevant.

## 🧪 STEP 2 — PROVE LOGS REACH SPLUNK (NO ANALYSIS YET)

### In Splunk (UI)

#### 1️⃣ Open **Search & Reporting**

Do **not** use dashboards.

#### 2️⃣ Set a **wide time range**

Select:

* **Last 24 hours** (not 15 minutes)

Why?  
Because Phase 0 distrusts default time windows.

#### 3️⃣ Minimal search (no logic)

Use only:

index=linux

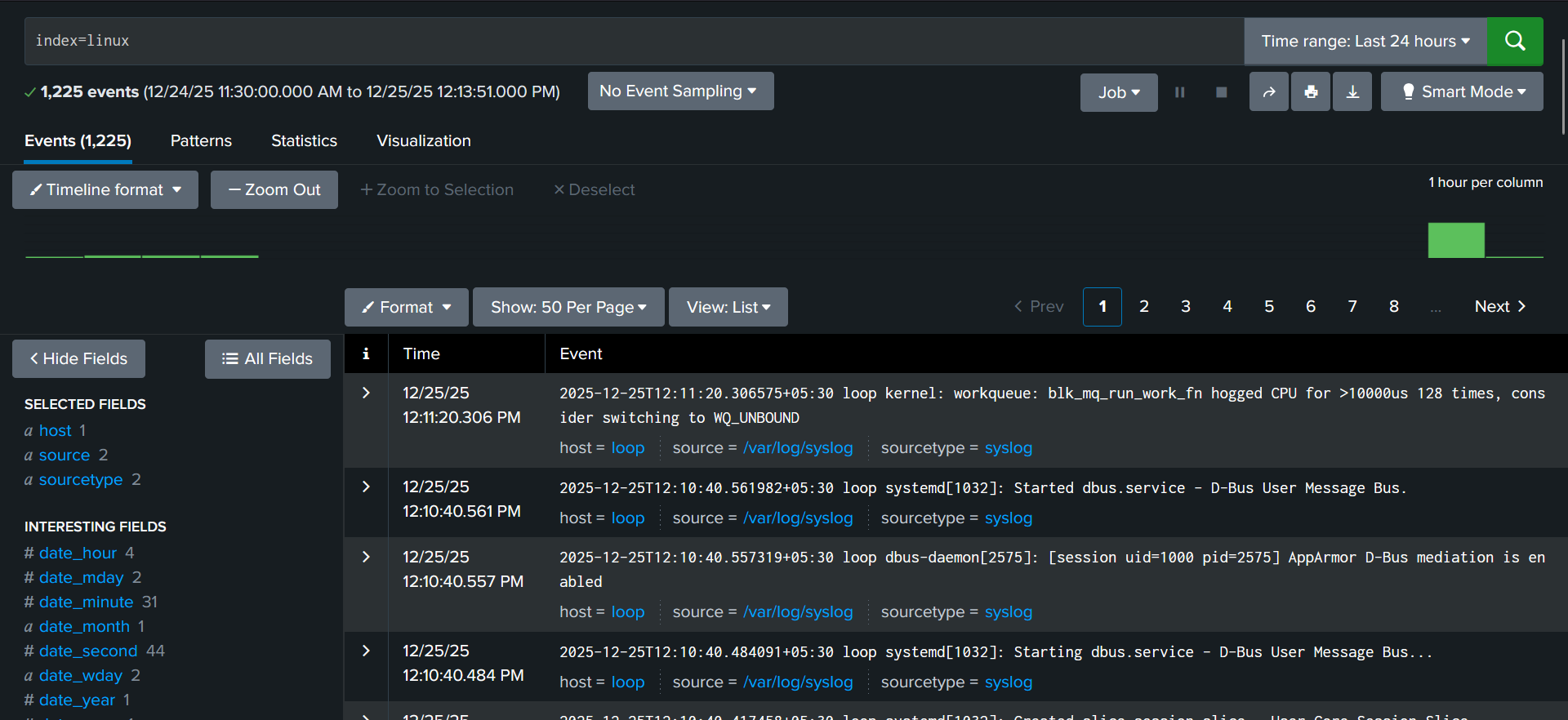
Do **not** add keywords yet.

### ✅ What you are checking

* Does data exist?
* Is data continuous?
* Are there obvious gaps?

If nothing appears:

* Don’t panic
* This is a **data trust finding**



## 🧪 STEP 3 — CONFIRM AUTH LOG EVENTS ARRIVE

Now narrow **slightly**.

Search:

index=linux auth

(or sshd if auth logs include it)

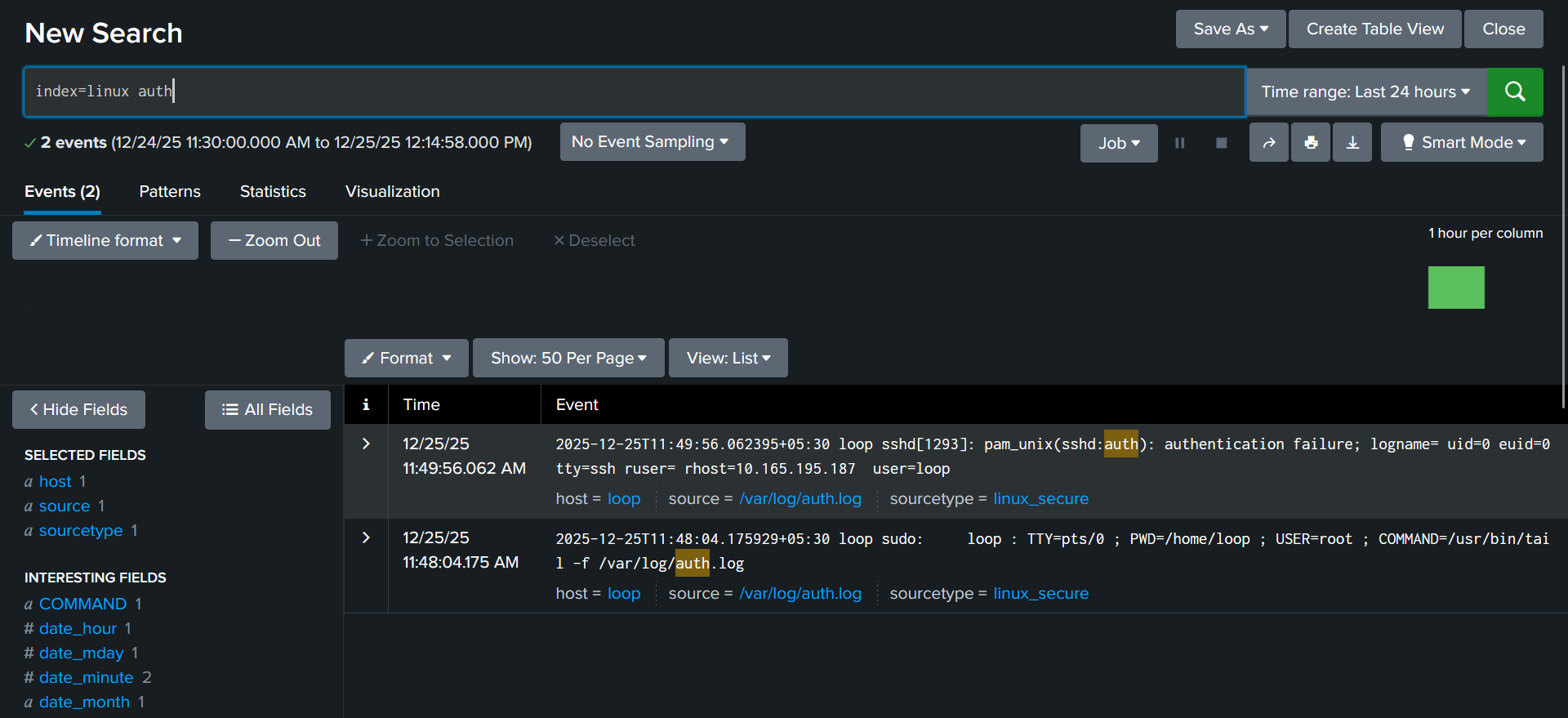
Do NOT assume fields yet.

### Observe:

* Do you see your SSH attempts?
* Do you see sudo activity?
* Do timestamps roughly match when you tested?

⚠️ If you don’t see your test events:

* Increase time range
* Do not assume failure yet



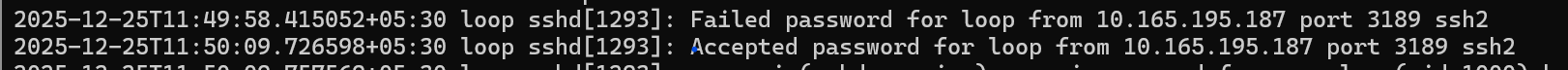
## 🧪 STEP 4 — COMPARE EVENT TIME VS SPLUNK \_time (CRITICAL)

Pick **one event** you just generated.

### 1️⃣ Note timestamp from Linux log

Example:

Dec 25 10:01:12



### 2️⃣ Click the same event in Splunk

Expand it and look at:

* \_time

### Compare:

* Is \_time equal to log time?
* Is there delay?
* Is delay consistent or random?

### ✅ Record this explicitly:

* “Event happened at 11:49:50”
* “Indexed at 11:49:56 ”
* “Delay =6 Seconds (Not a problem)”

This is **evidence**, not opinion.

## 🧪 STEP 5 — CHECK FOR DELAYED INGESTION PATTERNS

Still using a **wide time range**:

Look for:

* Flat gaps with no events
* Sudden bursts of old timestamps
* Events arriving out of order

Ask yourself:

Did Splunk receive this now, or did it happen now?

If unsure → widen time again.

## 🧪 STEP 6 — VERIFY PARSING & SOURCETYPE (MOST IMPORTANT)

Expand a Linux auth event.

### Check ****these 4 things****:

1️⃣ **Timestamp**

* Does it match the log?
* Or does it look like index time?

2️⃣ **Event structure**

* One log line = one event?
* Or multiple logs merged?

3️⃣ **Fields**  
Look for:

* user
* src / src\_ip
* host
* sourcetype
* source

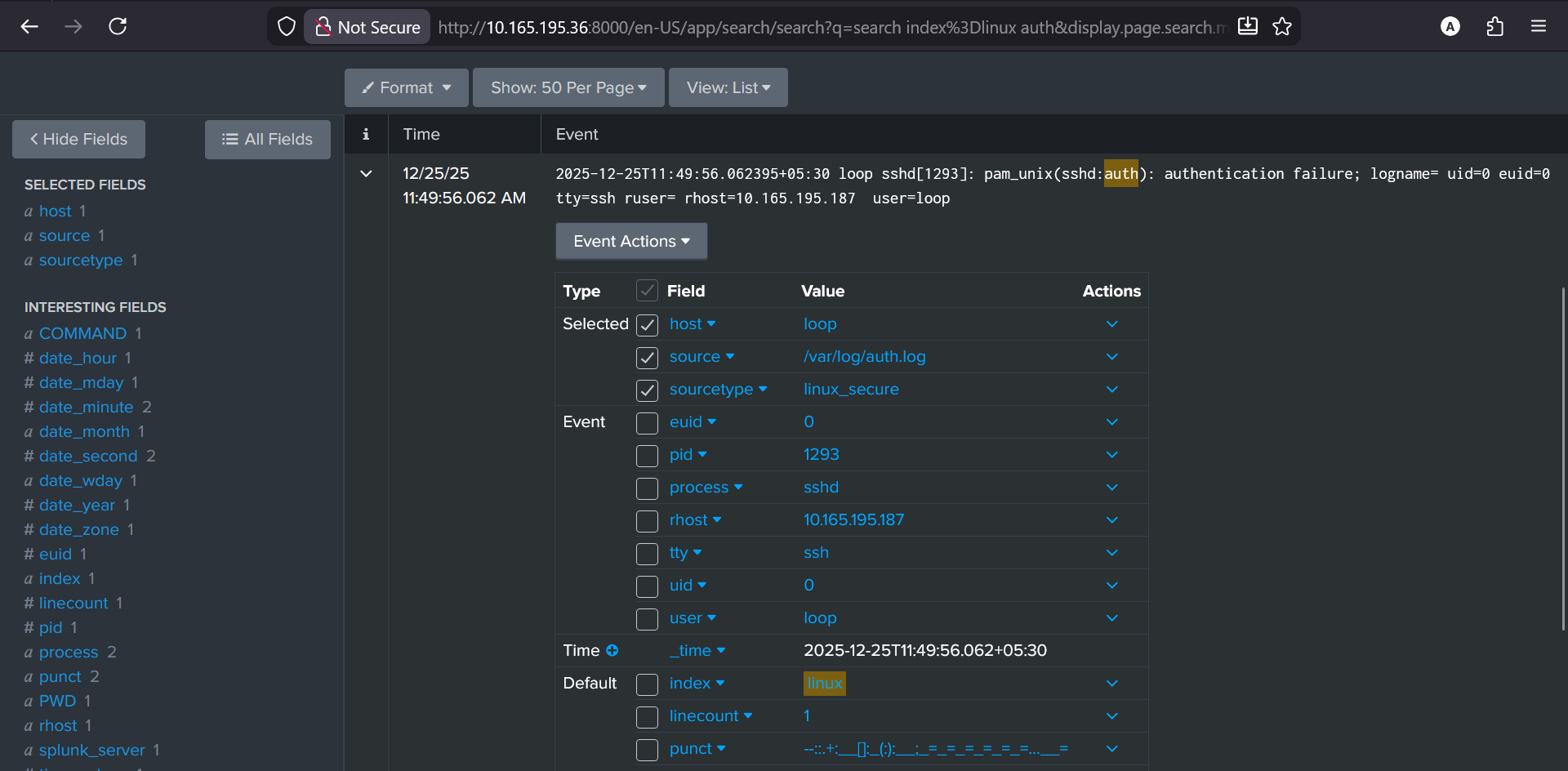
If these are missing → parsing is broken.

4️⃣ **Sourcetype**  
Note exactly:

sourcetype=\_\_\_\_

If sourcetype looks generic or wrong:

* That’s a **Phase 0 failure**
* Do not continue



## 🧪 STEP 7 — VERIFY EVENT BREAKING (MANUAL CHECK)

Take **one log line** from /var/log/auth.log.

Compare:

* That exact line
* To exactly one Splunk event

If:

* One log → multiple events ❌
* Multiple logs → one event ❌

Then:

Event breaking is broken → stop here.

Counts and timelines will lie.

## 🧠 STEP 8 — MAKE THE PHASE 0 DECISION (THIS IS IMPORTANT)

You must now answer **yes or no** to this question:

“If an SSH incident happened right now, could I reconstruct a **reliable timeline** using this data?”

### If ****YES****

* Data is trustworthy
* You may move to Phase A

### If ****NO****

* Identify which failed:
  + Generation
  + Ingestion
  + Time
  + Parsing
  + Event breaking

SOC maturity = knowing when **not** to proceed.

## 📝 WHAT I WANT FROM YOU

Reply with **observations**, not conclusions. Use this format:

1. Linux log generation: OK

2. Logs visible in Splunk: YES

3. Event time vs index time: (delay? no

4. Parsing status: OK

5. Event breaking: OK /

6. Can I trust this data for investigation? YES

I’ll review it like a senior analyst reviewing your shift notes — and only then will we move forward.

**🔥 PHASE A — Splunk Interface & Reality**

**Theory**

* What Search & Reporting *really* is (search UI, not SOC)
* Raw events vs extracted fields
* Time picker lies and limitations

**Practical**

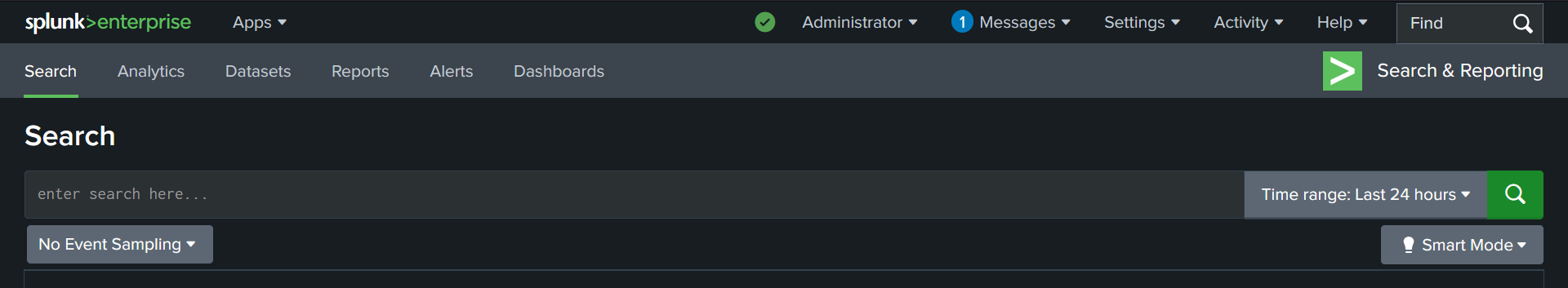
**🧪 STEP 1 — OPEN SEARCH & REPORTING (NO SEARCH YET)**

1. Log in to Splunk
2. Open **Search & Reporting**
3. **Do not type anything**

**Just observe the screen:**

* Time picker (top right)
* Event count
* Fields sidebar
* Raw events panel

📌 **Write down**

* What is the default time range?
* 
* Does it *look* like a complete view? No

This is your **baseline illusion**.

**🧪 STEP 2 — RUN THE MOST BASIC SEARCH POSSIBLE**

Now type **only** this:

index=linux

Do NOT add:

* sshd
* auth
* sudo
* filters

Run the search.

**Observe (do not judge):**

* Number of events
* Are events continuous or sparse?
* Any visible gaps?

📌 Write down:

* Event count: 3,727 events
* First visible timestamp : Dec 1, 2025
* Last visible timestamp: Jan, 1 2025



**🧪 STEP 3 — CHANGE TIME RANGE: 15 MINUTES**

1. Set time picker to **Last 15 minutes**
2. Run the same search again:

index=linux

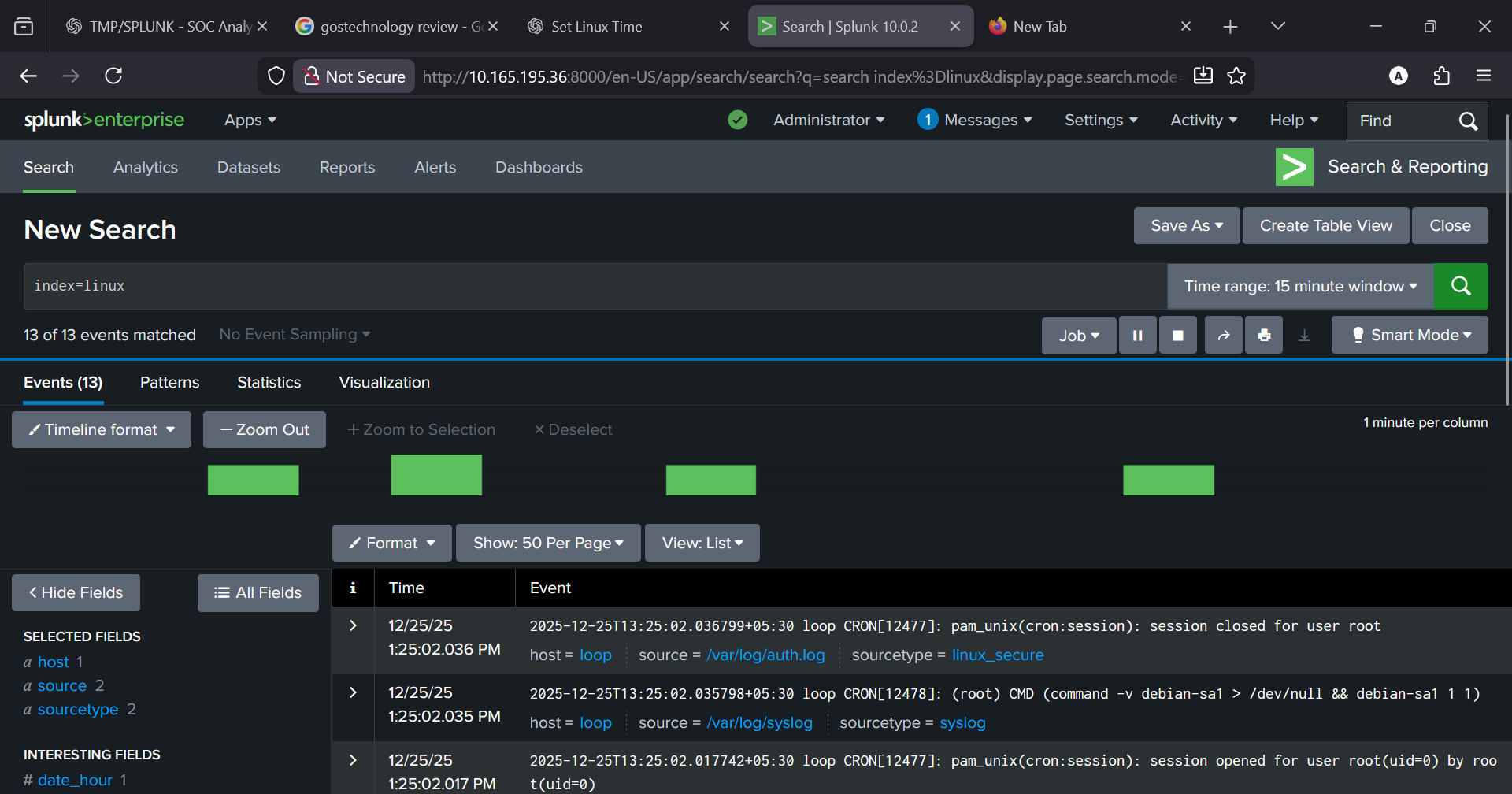
**Observe:**

* Did event count drop?
* Did some hosts disappear?
* Did activity look “quiet”?

📌 Write down:

* Event count (15m) : 13
* Oldest event time shown 12/25/25 ~1:25 PM

🚨 **Do NOT conclude “nothing happened”**



**🧪 STEP 4 — CHANGE TIME RANGE: 1 HOUR**

Now:

1. Change time picker to **Last 1 hour**
2. Run the **same query**

**Observe:**

* New events appear?
* Gaps fill in?
* Bursts visible?

📌 Write down:

* Event count (1h) : 53
* Any gaps you notice : yes

**🧪 STEP 5 — CHANGE TIME RANGE: 24 HOURS (CRITICAL STEP)**

Now:

1. Change time picker to **Last 24 hours**
2. Run:

index=linux

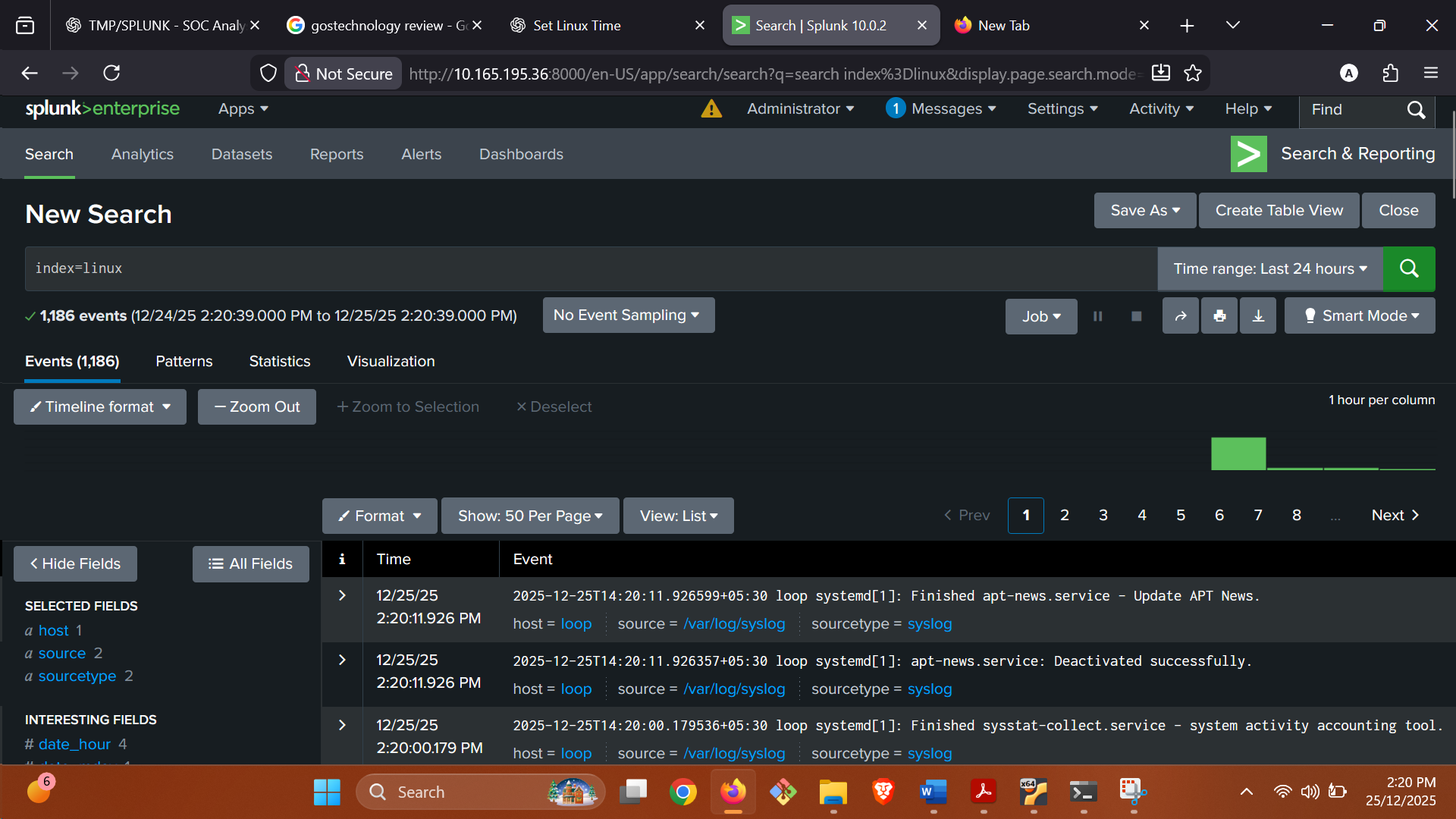
**Observe carefully:**

* Events that were invisible before
* Large bursts of events
* Long quiet gaps
* Events with **older timestamps appearing later**

📌 Write down:

* Event count (24h) : 1,186
* Any sudden jumps in volume : yes
* Any out-of-order timestamps : No

This step usually **breaks blind trust**.



**🧪 STEP 6 — RAW EVENTS vs FIELDS (DO NOT SKIP)**

1. Click **one event**
2. Expand it

**Look at:**

* **Raw event text** (the log line)
* **Extracted fields** (user, host, source, etc.)

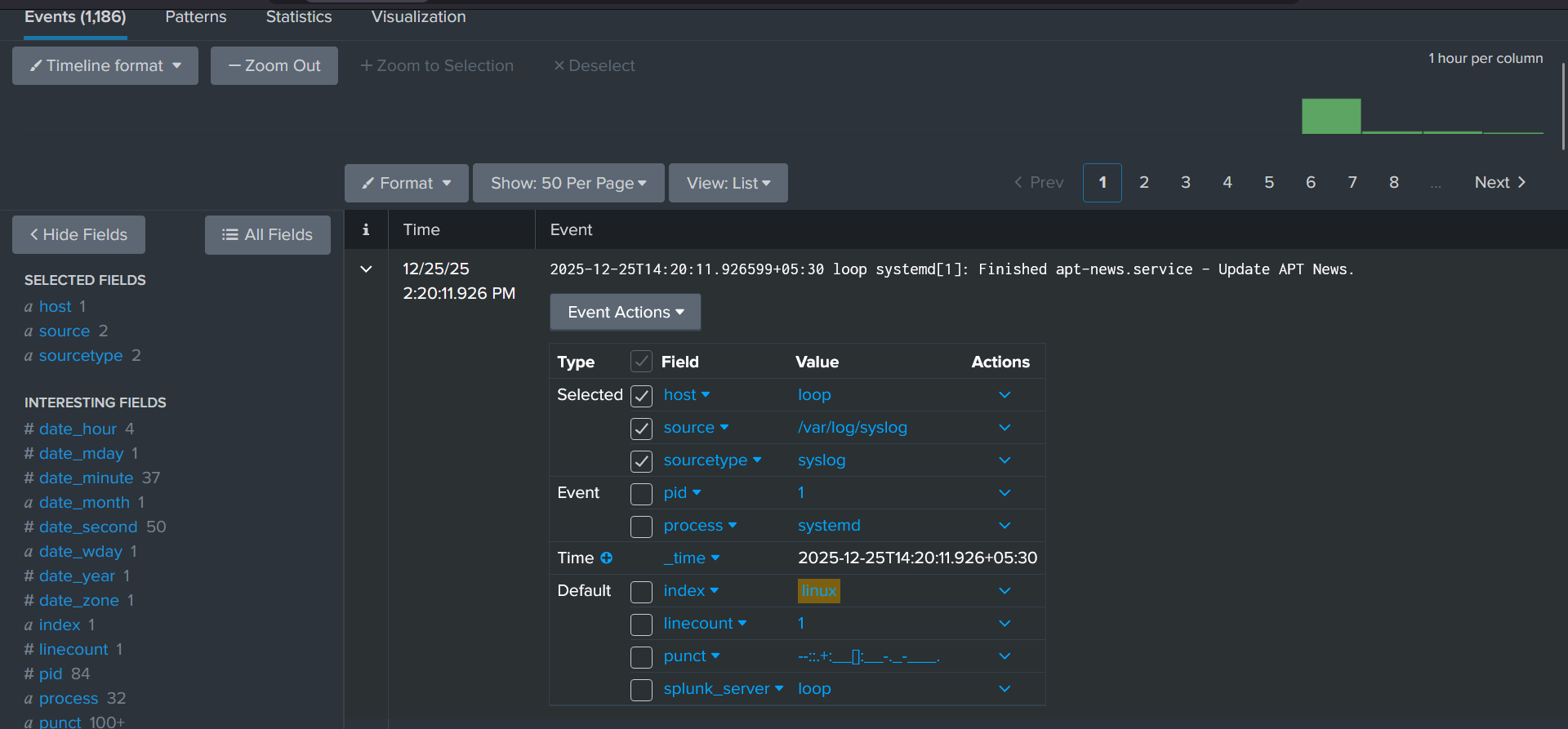
Now ask:

* Does raw text clearly show what happened?
* Do fields correctly represent that text?

📌 Write down:

* One field that matches raw text
* One thing raw text shows that fields hide (if any)

**Matching field:** process=systemd matches systemd[1] in raw log  
**Hidden detail:** Raw text shows apt-news.service finished successfully, which is not fully represented as structured fields



**🧪 STEP 7 — TIME PICKER ILLUSION TEST**

Do this carefully:

1. Pick an event with timestamp **X : 14:20:11**
2. Narrow time picker **slightly smaller than X : Last 5 minutes**
3. Run search again

**Observe:**

* Did the event disappear? Yes
* Did it *still exist* in reality? Yes

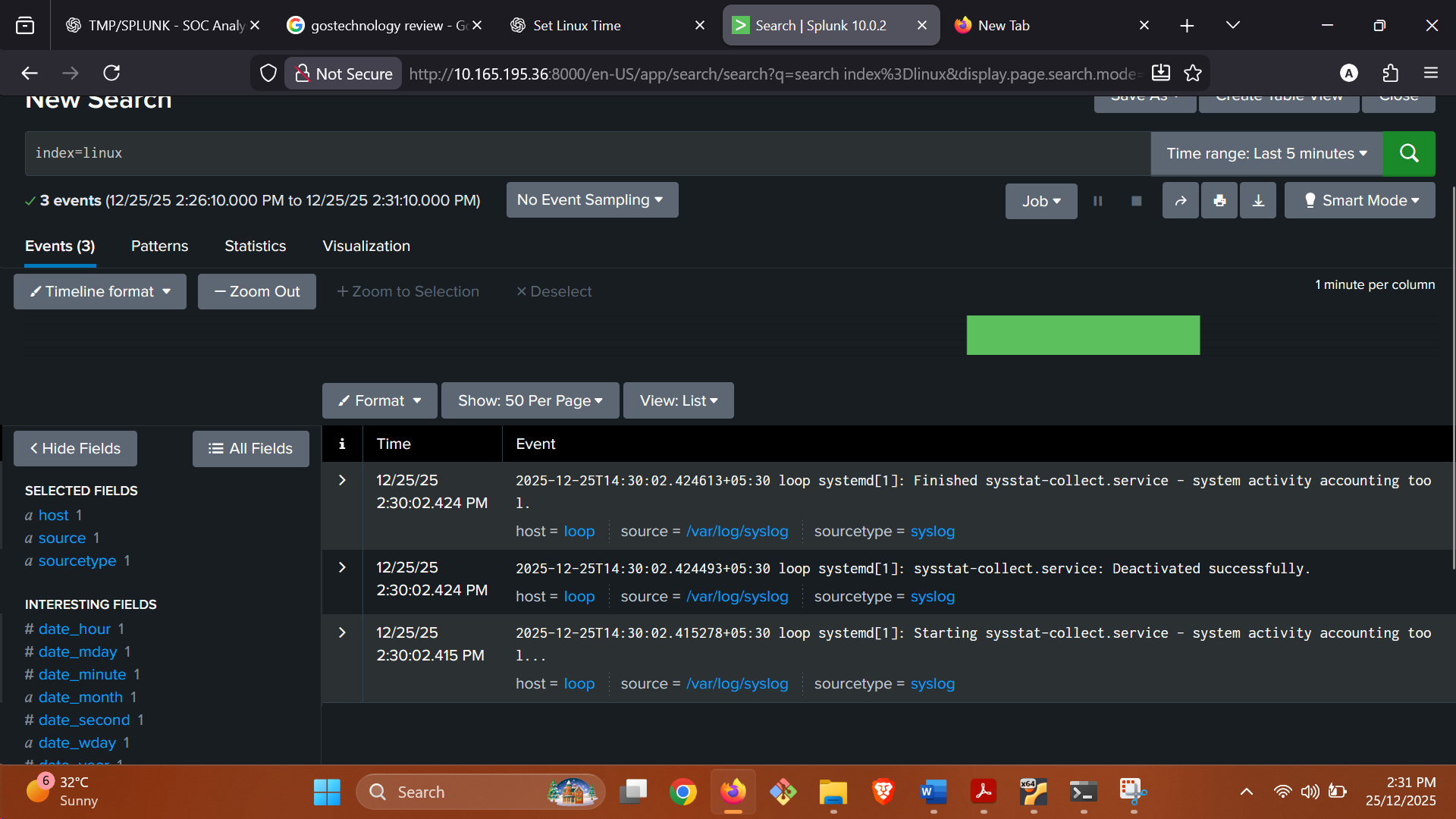
📌 Write down:

* What disappeared when time was narrowed

Previously observed events (older timestamps) disappeared from the search results when the time picker was narrowed.

* What this implies about UI trust

Event visibility in the Splunk UI depends on the selected time range; disappearance from view does not mean the data is missing or deleted.



**📝 WHAT I WANT FROM YOU**

1. Default time range was: 24hour

2. Events in 15 minutes: 13

3. Events in 1 hour: 53

4. Events in 24 hours: 1,186

5. New things visible only in 24h view:

* System-wide background activity (cron jobs, systemd services, APT updates)
* Burst patterns across different hours
* Quiet periods between bursts that were invisible in shorter views

6. Did any events appear delayed or out of order? NO

7. One difference between raw event and fields:

* **Raw event** shows full context (e.g., service name and action like apt-news.service finished)
* **Fields** only summarize parts (e.g., process=systemd) and hide detailed intent

**🔥 PHASE B — Searching (QUESTION-FIRST)**

**Theory**

* Searches answer **investigative questions**
* Keywords are weak, fields are strong

**🔥 PHASE B — SEARCHING (QUESTION-FIRST) — PRACTICAL**

**Objective:** Learn how searches answer **investigative questions**, and why **keywords explore** while **fields prove**.

**🧠 RULES FOR THIS PHASE**

* ❌ No alerts, no dashboards
* ❌ No “this looks malicious”
* ✅ One question → one search
* ✅ Observe before narrowing

**STEP 0 — WRITE THE QUESTION (MANDATORY)**

**Question:**

“Is there SSH-related activity in Linux logs during my test window?”

Do **not** type anything until the question is written.

**STEP 1 — BASELINE SEARCH (DATA PRESENT?)**

**Search**

**index=linux**

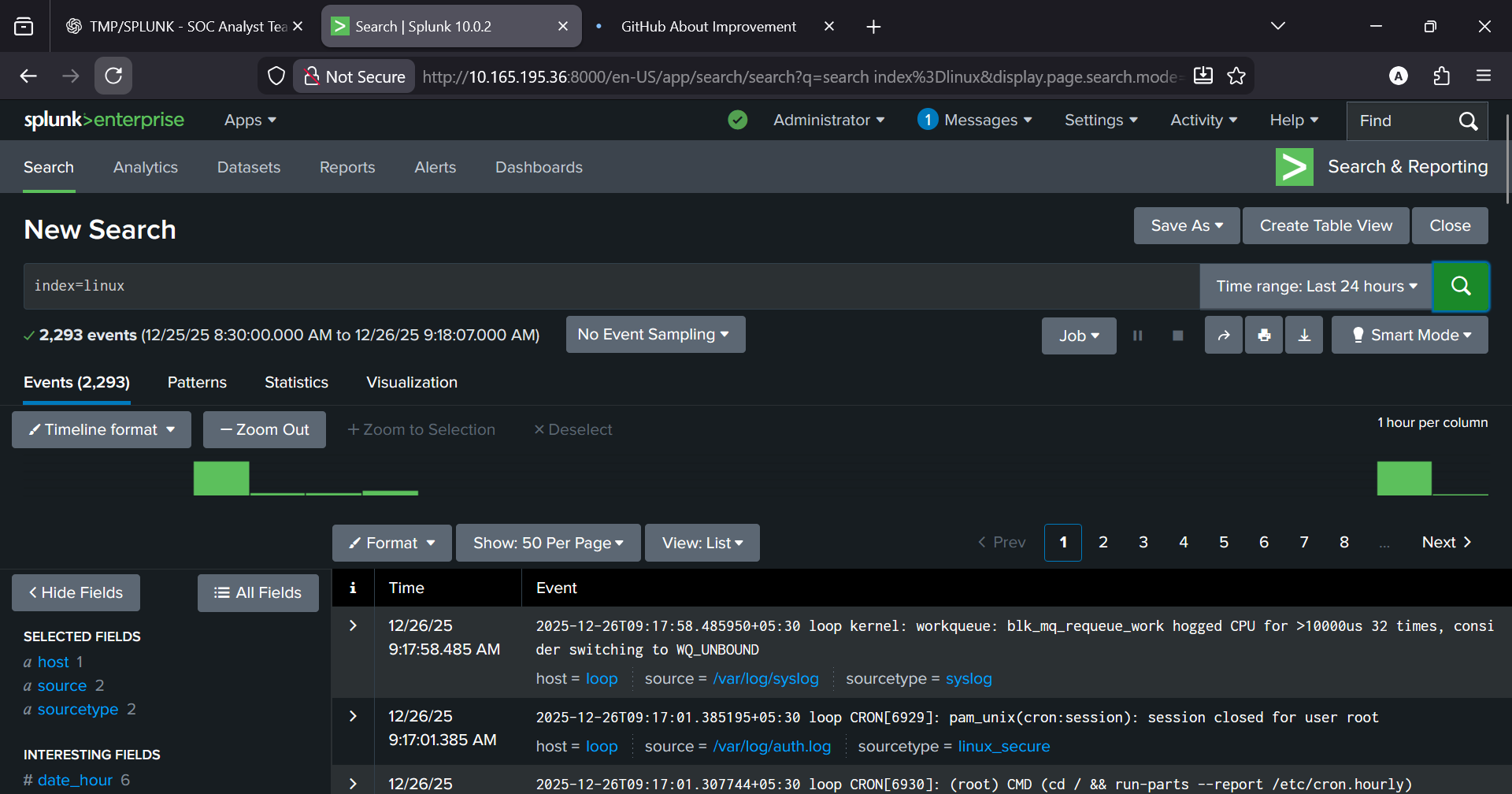
**Why this exists:**

“Do I even have Linux data right now?”

**Observe**

* Event count : 2,293 events
* Time span shown : 12/25/25 8:30:00 AM → 12/26/25 9:18:07 AM
* Are events continuous or sparse? This is **continuous enough** to investigate

**Do not** add filters yet.



**STEP 2 — KEYWORD SEARCH (EXPLORATION)**

**Search**

index=linux sshd

**Question it answers:**

“Do any logs mention SSH activity textually?”

**Observe**

Event count vs Step 1 **STEP 1:** 2,293 events

**STEP 2:** 17 events

* Hosts appearing : host = loop
* Are events clustered or spread? : Events are CLUSTERED

⚠️ **Do not conclude** anything about attacks.

**STEP 3 — SECOND KEYWORD (COMPARE CONTEXT)**

**Search**

index=linux sudo

**Question it answers:**

“Is sudo activity logged and visible?” Yes

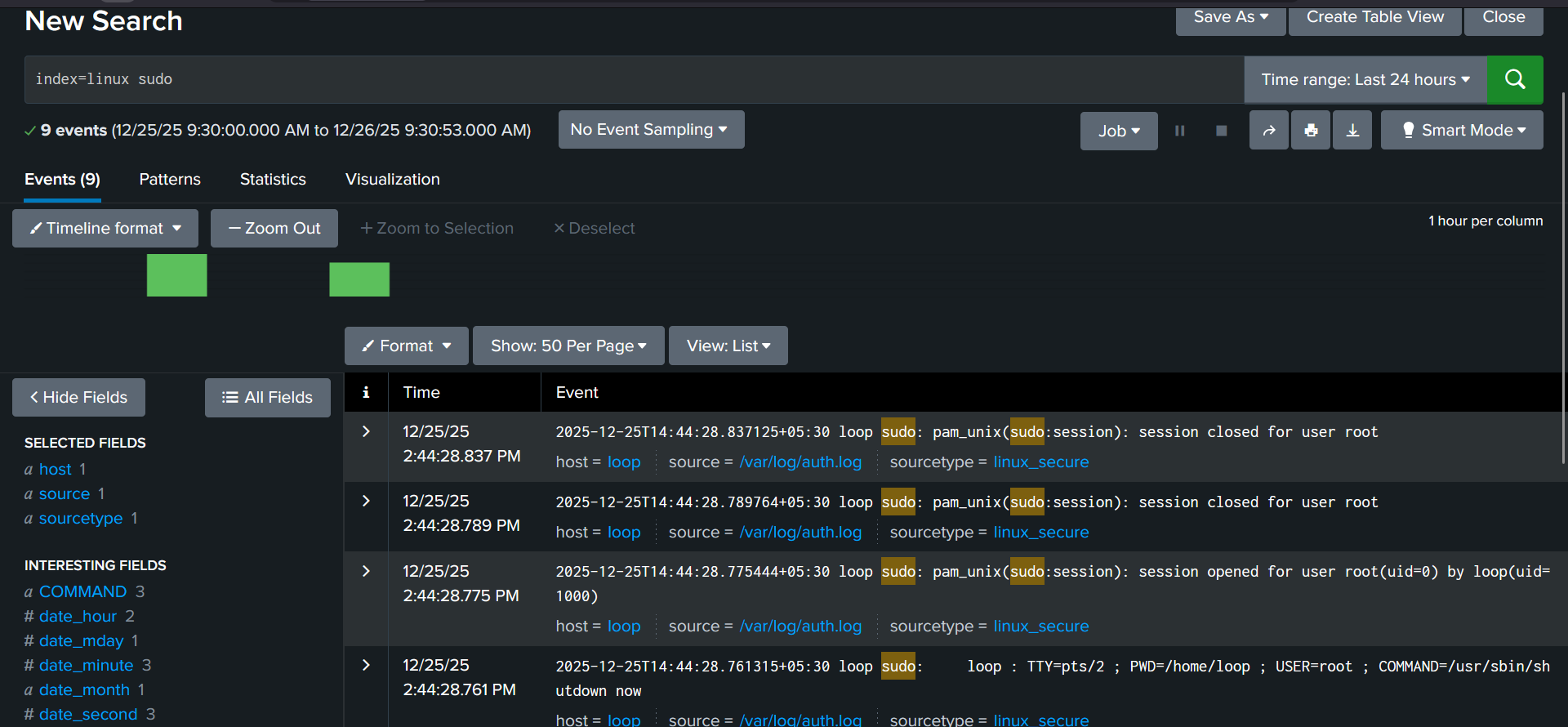
**Observe**

Compare volume with sshd ( sshd → **17 events**

sudo → **9 events)**

* Do sudo events appear around the same times? No

You are learning **context**, not verdicts.



**STEP 4 — INSPECT RAW EVENT (GROUND TRUTH)**

Click **one SSH-related event** from Step 2.

**Look at**

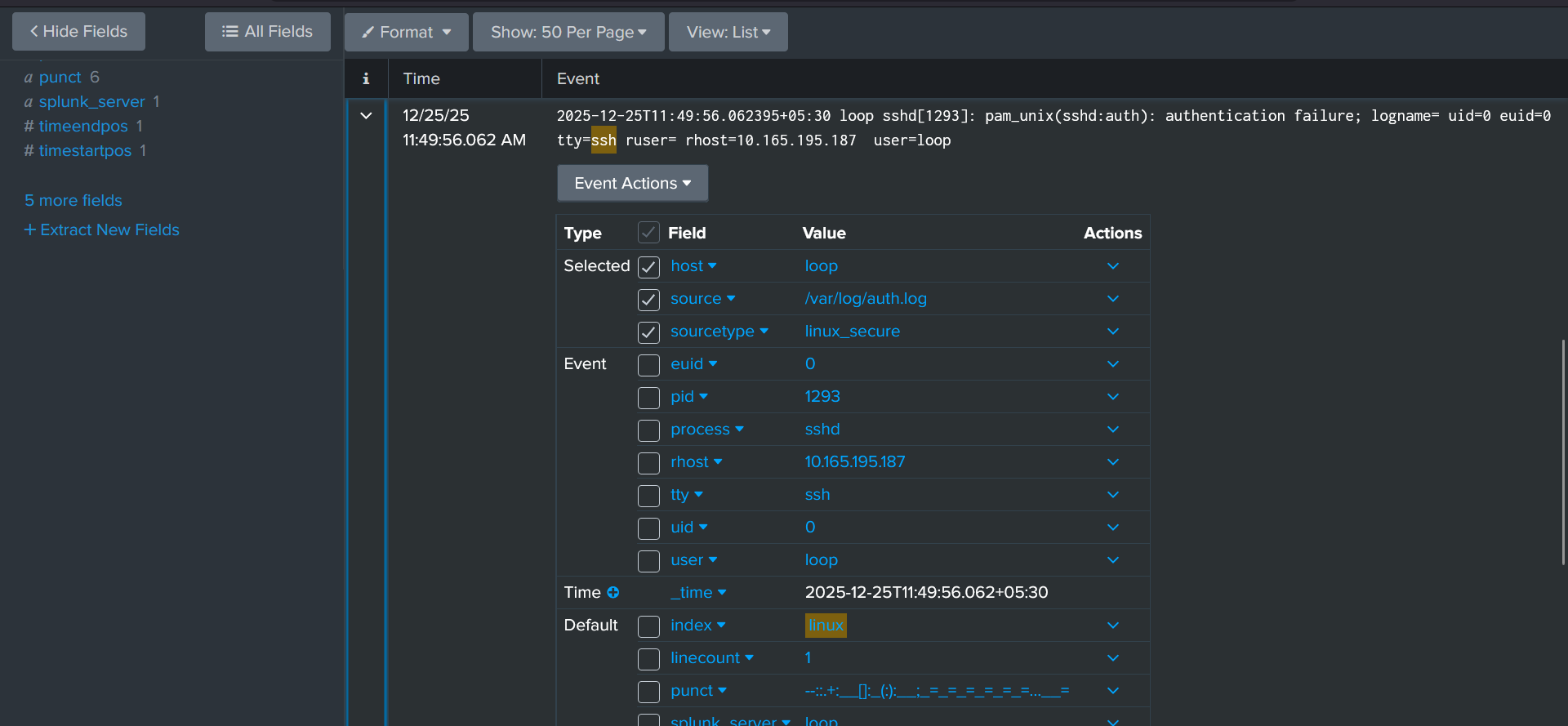
* Raw event text (the full log line)
* The timestamp inside the log
* Words that indicate success/failure

**Write down**

* One fact the raw text shows clearly

This is a FAILED SSH authentication attempt.

Senior habit: raw text is the last court of appeal.



**STEP 5 — IDENTIFY AVAILABLE FIELDS (NO FILTERING YET)**

With the same event expanded, look at the **Fields panel**.

**Identify**

* Host: Loop
* Source: /var/log/auth.log
* Sourcetype: linux\_secure

Any user-related field: **user = loop**

**uid = 0**

**euid = 0**

* Any IP-related field: rhost ip: 10.165.195.187

**Do NOT filter yet.**  
Just confirm fields exist and make sense.

**STEP 6 — FIELD-BASED NARROWING (STABILITY TEST)**

Now we move from **explore** → **prove**.

**Search**

index=linux sourcetype=<the\_sourcetype\_you\_saw>

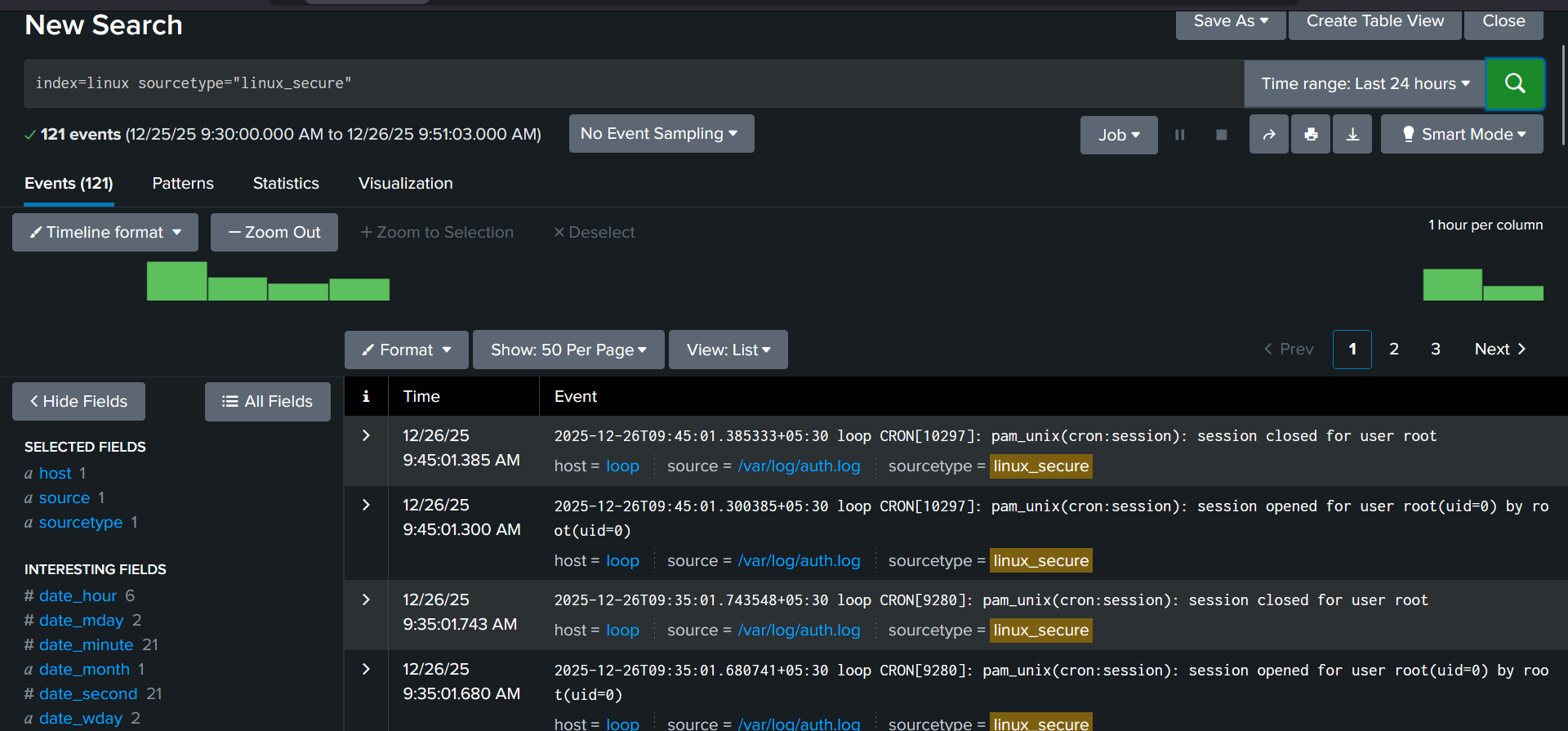
(Replace with the exact sourcetype you observed.)

**Question it answers:**

“Show me Linux auth events based on structure, not text.”

**Observe**

* Event count stability : 121
* Do SSH events still appear? Yes
* Does volume change less randomly than keyword search? YES



**STEP 7 — CONTROLLED COMPARISON (KEYWORD vs FIELD)**

Run these **back-to-back**:

**A. Keyword**

index=linux sshd

**B. Field**

index=linux sourcetype=linux\_secure

**Observe**

* Which one is more consistent? Field-based search
* Which one would you defend in an incident report? Field-based search

👉 This is why SOC prefers **fields**.

**STEP 8 — NARROW WITH PURPOSE (ONE FILTER ONLY)**

**New Question:**

“Which hosts show SSH-related activity?”

**Search**

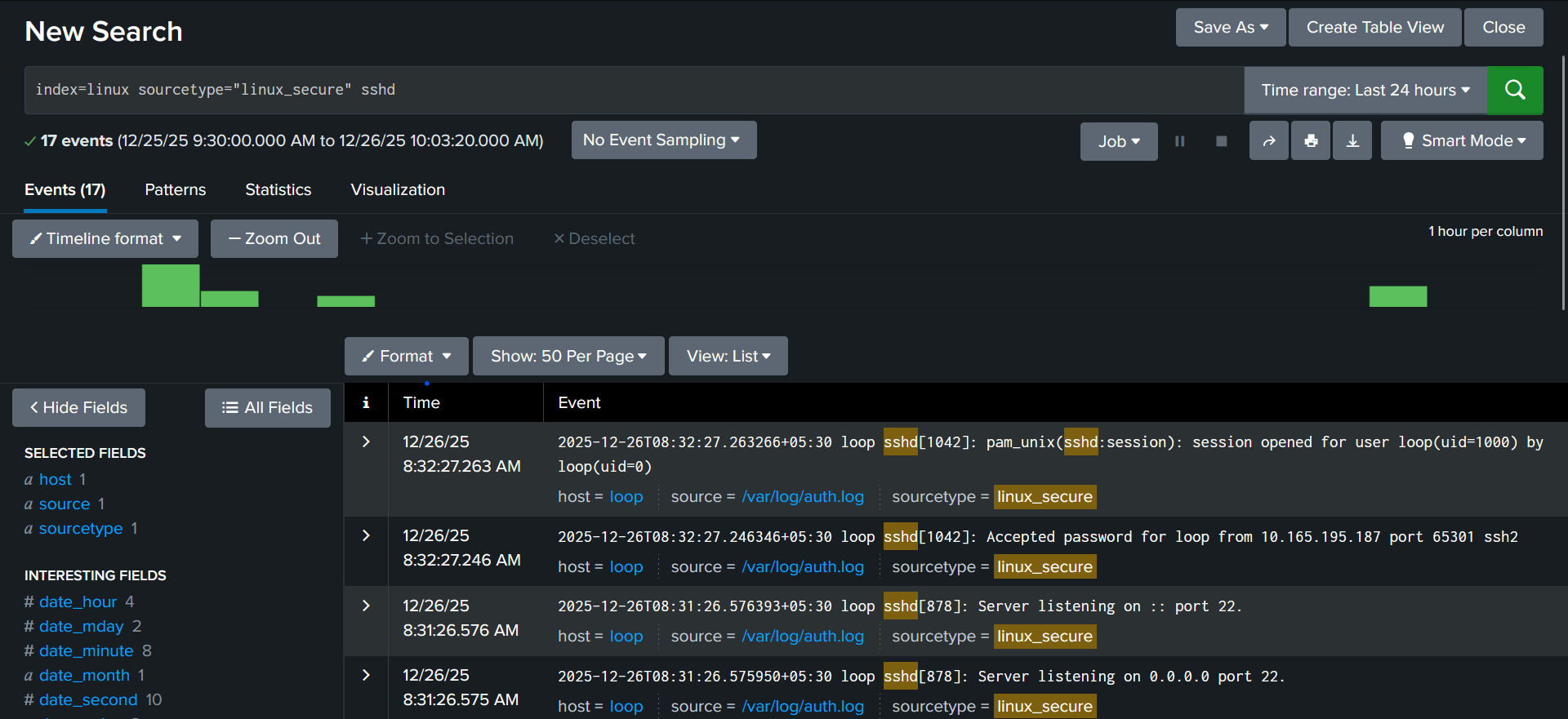
index=linux sourcetype=linux\_secure sshd

(One keyword, one structural filter.)

**Observe**

* Hosts involved : host = loop
* Are multiple hosts present? NO

Stop here. No stacking filters.



**STEP 9 — CHECK TIME DEPENDENCE (DON’T TRUST UI)**

Run **the same search** with:

* Last 15 minutes
* Last 1 hour
* Last 24 hours

**Observe**

* What disappears?

1. SSH login events
2. Earlier auth failures/successes
3. Context needed for sequences

* What only appears when you widen time?
* SSH activity
* Authentication patterns
* Event sequences (fail → success, service start → login)

This teaches you **why defaults lie**.

**STEP 10 — PHASE B SELF-CHECK (YES/NO)**

Answer honestly:

* Did you write a question before each search?
* Did keywords help explore but not prove?
* Did field-based searches feel more stable?
* Did widening time change conclusions?

If “yes” to all → Phase B passed.

**📝 WHAT TO SEND ME NEXT**

Reply with this exact format:

1. **Question I started with:**  
   Is there SSH-related activity in Linux logs during my test window?
2. **Event count (index=linux):**  
   ~2,293 events (24-hour baseline)
3. **Event count (sshd keyword):**  
   17 events
4. **Event count (sudo keyword):**  
   9 events
5. **Sourcetype observed:**  
   linux\_secure
6. **One useful field found:**  
   rhost (remote source IP)
7. **Keyword vs field felt more reliable because:**  
   Field-based searches rely on log structure rather than fragile text matching, making results more consistent and defensible.
8. **One thing that appeared only after widening time:**  
   SSH authentication activity that was invisible in narrow time windows

**🔥 PHASE C — FIELDS (EVIDENCE LAYER)**

**Theory**

* Fields = defensible evidence
* Raw text = supporting detail only

**🔥 PHASE C — FIELDS (EVIDENCE LAYER) — PRACTICAL**

**Objective:** Learn to rely on **fields** for conclusions and use **raw text** only to confirm meaning.

**🧠 RULES FOR THIS PHASE**

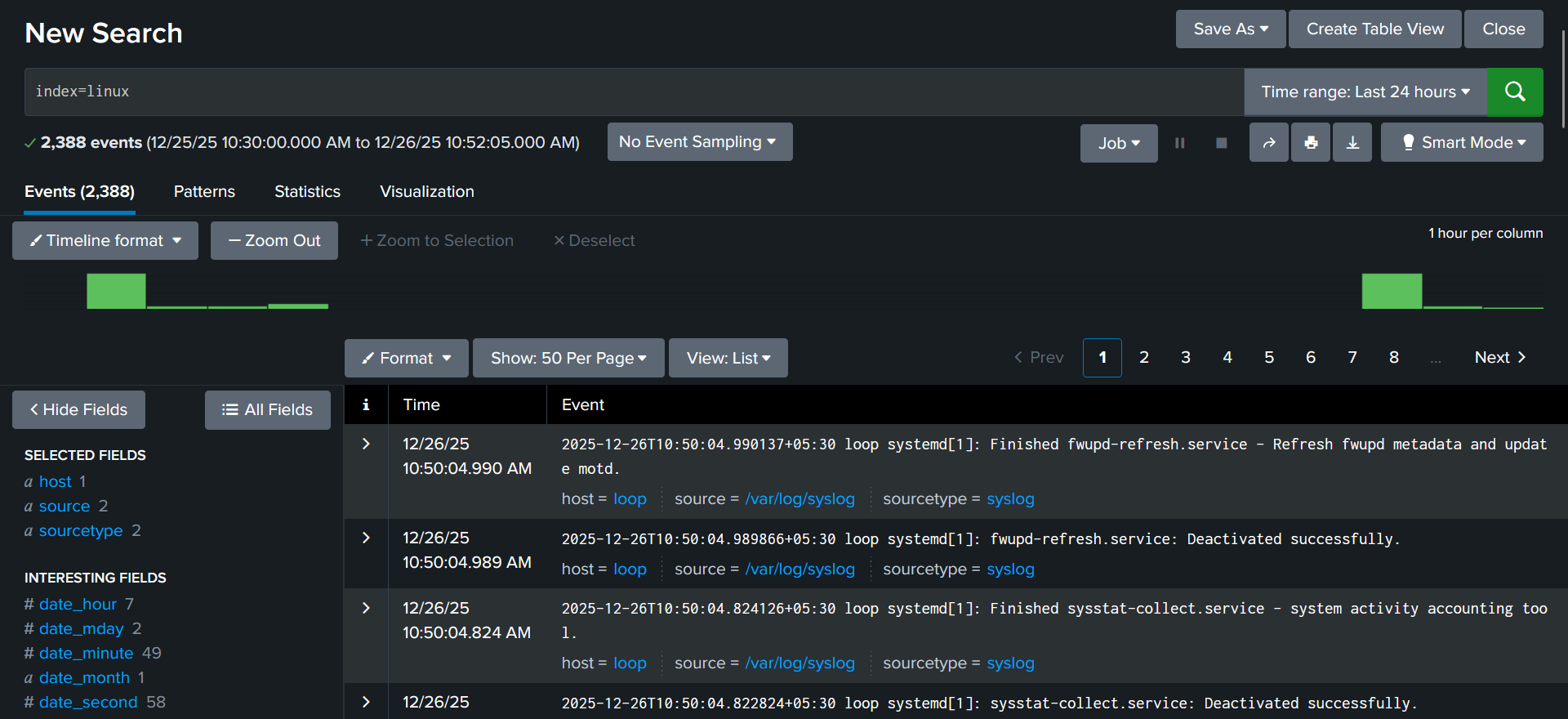
* ❌ No alerts, no dashboards
* ❌ No scrolling-as-analysis
* ✅ Expand events and read fields
* ✅ One question at a time

**STEP 1 — OPEN A SAFE BASE VIEW**

1. Open **Search & Reporting**
2. Set **Time Range = Last 24 hours**
3. Run:

index=linux

**Observe only:** data exists, time span looks reasonable.



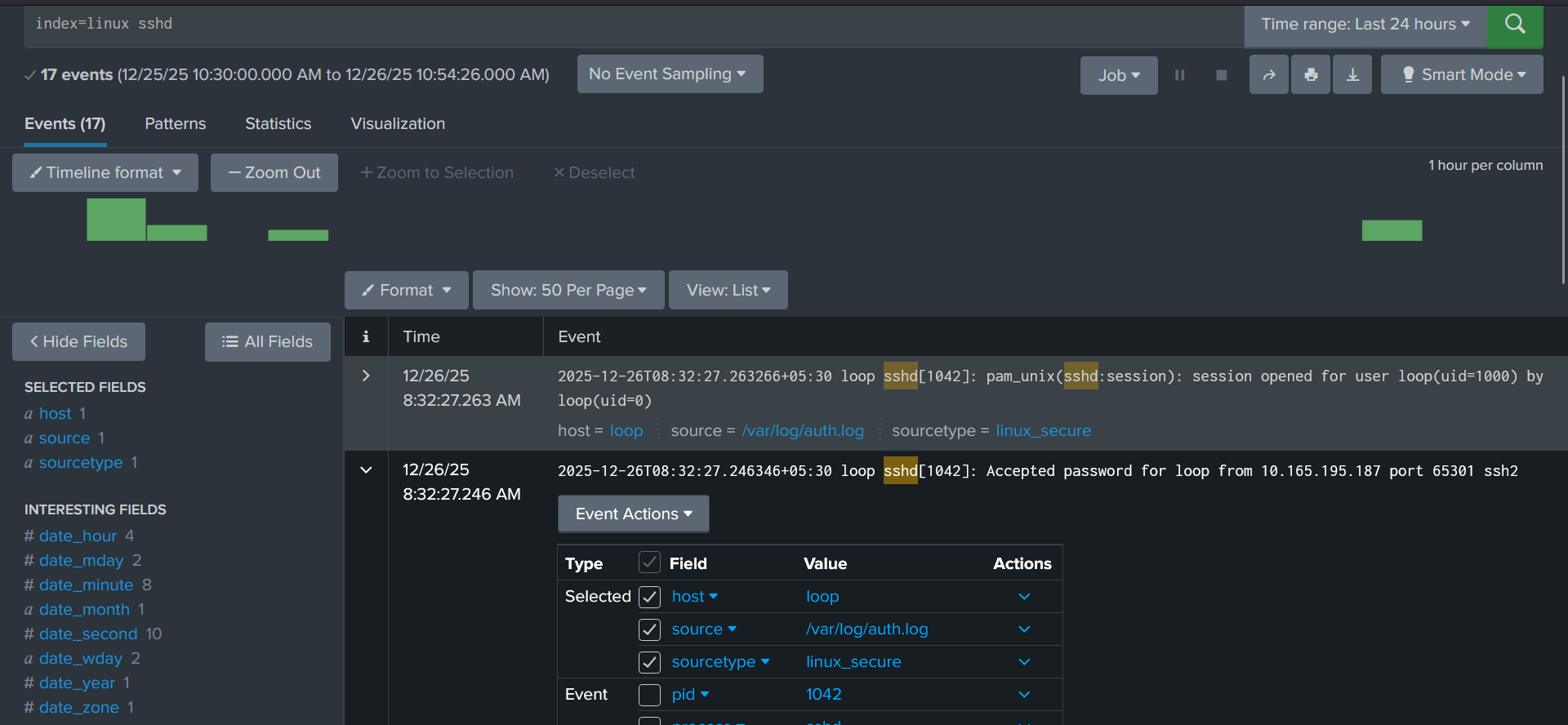
**STEP 2 — FIND A LINUX AUTH EVENT**

Narrow just enough to locate auth-related logs:

index=linux sshd

(or sudo if that’s more common in your environment)

Click **one event** that clearly looks like auth activity.



**STEP 3 — EXPAND THE EVENT (GROUND TRUTH)**

Fully expand the event.

You must see two areas:

* **Raw event** (original log line)
* **Fields panel** (key=value pairs)

Do **not** filter yet.

**STEP 4 — IDENTIFY MANDATORY INFRA FIELDS**

In the **Fields panel**, explicitly find and note:

* host → which machine : loop
* source → which file/input : /var/log/auth.log
* sourcetype → how Splunk parsed it : linux\_secure

**Checkpoint:**  
If any of these are missing → stop (Phase 0 failure).

**STEP 5 — IDENTIFY USER-RELATED FIELDS**

Still in the same event, look for **identity** fields. Names vary.

Details:

* **user: loop**
* **user\_name : loop**
* **uid: 1000**

**Write down exactly what exists** in *your* data. Don’t guess names.

**STEP 6 — IDENTIFY IP-RELATED FIELDS**

Look for **source IP** fields. Common variants:

* src
* src\_ip
* client\_ip
* rhost: 10.165.195.187

Again, note what actually exists.

## What does ****NOT**** exist (and that’s normal)

You will **not** see:

* src
* src\_ip
* client\_ip

Why:

* Those field names are common in **network devices**, **cloud logs**, or **Windows**
* Linux SSH (linux\_secure) typically uses **rhost**

**STEP 7 — MAP RAW TEXT → FIELDS (CRITICAL)**

**Raw event text (core part)**

sshd[1042]: Accepted password for loop from 10.165.195.187 port 65301 ssh2

Now we **map facts in raw text → actual fields that exist**.

**Raw text → Field mapping (your environment)**

|  |  |  |
| --- | --- | --- |
| Raw text fragment | What it means | Actual field |
| sshd | Service handling auth | process=sshd (or visible via source / message) |
| Accepted password | Auth result = success | **In raw text only** (no dedicated success field) |
| loop | Username | user=loop |
| 10.165.195.187 | Remote source IP | rhost=10.165.195.187 |
| /var/log/auth.log | Log file | source=/var/log/auth.log |
| Event time text | When it happened | \_time |

**Ask:**

Do the fields faithfully represent the raw text?

If not → parsing is lying.

**STEP 8 — STATE THE QUESTION (MANDATORY)**

Write this question **before searching**:

**“Which users attempted SSH authentication?”**

Don’t type yet.

**STEP 9 — TRY TO ANSWER WITH RAW TEXT ❌**

Run:

**index=linux sshd**

Now try to answer the question by **scrolling**:

* Which users?
* How many per user?

**Notice:** you eyeball, you guess, you can’t defend it.

That’s the lesson.

**STEP 10 — ANSWER THE SAME QUESTION WITH FIELDS ✅**

Use **structure**, not text.

Start with the auth structure you observed:

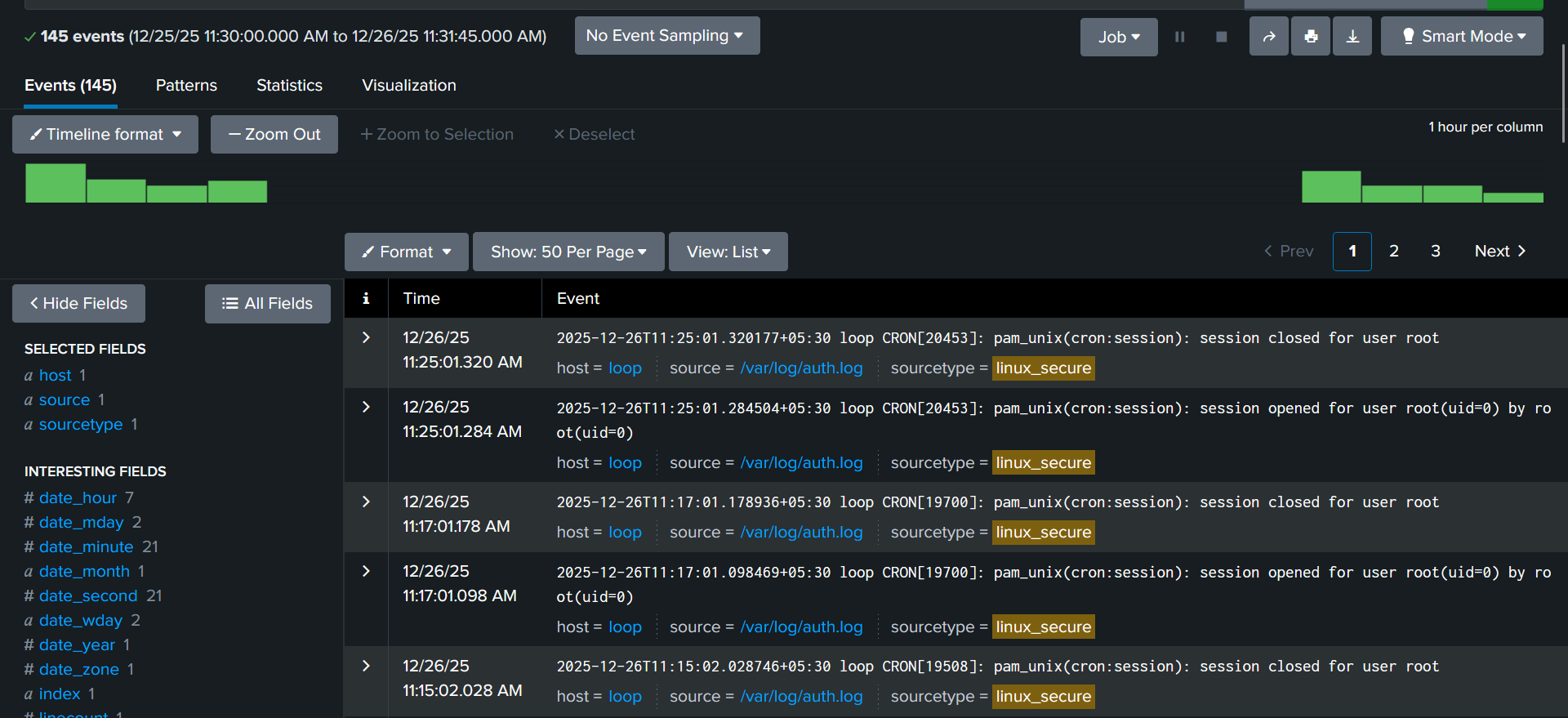
**index=linux sourcetype=linux\_secure**

Now **filter using the user field** you identified (via the Fields sidebar, click it):

* This ensures you’re using **actual fields**, not guessing syntax.

**Observe:**

* Clear user list
* Stable results
* Easy comparison



**STEP 11 — PROVE DEFENSIBILITY**

Now narrow by **one more field** (e.g., IP) using the Fields sidebar.

Ask:

Can I now say **who**, **from where**, and **how often**—without scrolling?

If yes → you’re doing it right.

**STEP 12 — FINAL SELF-CHECK (YES/NO)**

Answer honestly:

* Did I identify fields before filtering?
* Did fields match raw text?
* Could I answer the question without scrolling?
* Would this survive a review?

If all **YES** → Phase C passed.

**📝 WHAT TO SEND ME NEXT**

* **Sourcetype observed:**  
  linux\_secure
* **Mandatory fields present (host/source/sourcetype):**  
  YES
* **User-related field(s):**  
  user, uid
* **IP-related field(s):**  
  rhost
* **One example where raw text matched fields:**  
  Raw text showed “Accepted password for loop from 10.165.195.187” and the fields showed user=loop and rhost=10.165.195.187.
* **Why raw text alone failed to answer the question:**  
  Scrolling raw logs required guessing and eyeballing, making it impossible to reliably identify all users or count events per user in a defensible way.
* **Why fields succeeded:**  
  Fields provided structured, machine-countable data (e.g., user, rhost) that allowed clear identification, grouping, and comparison without ambiguity

**🔥 PHASE D — FILTERING (SCOPE CONTROL)**

**Theory**

* Filtering is **investigation narrowing**, not detection
* Precision reduces analyst bias

**🔥 PHASE D — FILTERING (SCOPE CONTROL) — PRACTICAL**

**Objective:** Learn how deliberate filtering changes the investigation narrative without “detecting” anything.

**🧠 HARD RULES (DO NOT BREAK THESE)**

* ❌ One filter at a time
* ❌ No stacking filters
* ❌ No conclusions
* ✅ Always return to baseline
* ✅ Observe before judging

If you stack filters, you lose perspective.

**STEP 0 — ESTABLISH THE BASELINE (MANDATORY)**

**1️⃣ Open Search & Reporting**

* Time range: **Last 24 hours**

**2️⃣ Run the stable base search**

**index=linux sourcetype=<your\_auth\_sourcetype>**

**What this answers**

“What does *all* Linux authentication activity look like?”

**Observe only:**

* Event count
* Time spread
* Variety of users, hosts, services

📌 **Write one sentence** describing the *overall noise level*.  
This is your reference point.

Linux authentication activity is steady and low-risk, dominated by predictable root cron sessions on a single host, with no bursty or multi-user patterns observed.



**STEP 1 — FILTER BY A SINGLE USER (USER PERSPECTIVE)**

**1️⃣ Locate the user field**

In the **Fields sidebar**, find:

* user / account / user\_name (whatever exists in *your* data)

**2️⃣ Click one specific user**

Splunk will auto-add the filter.

**Observe (do not judge):**

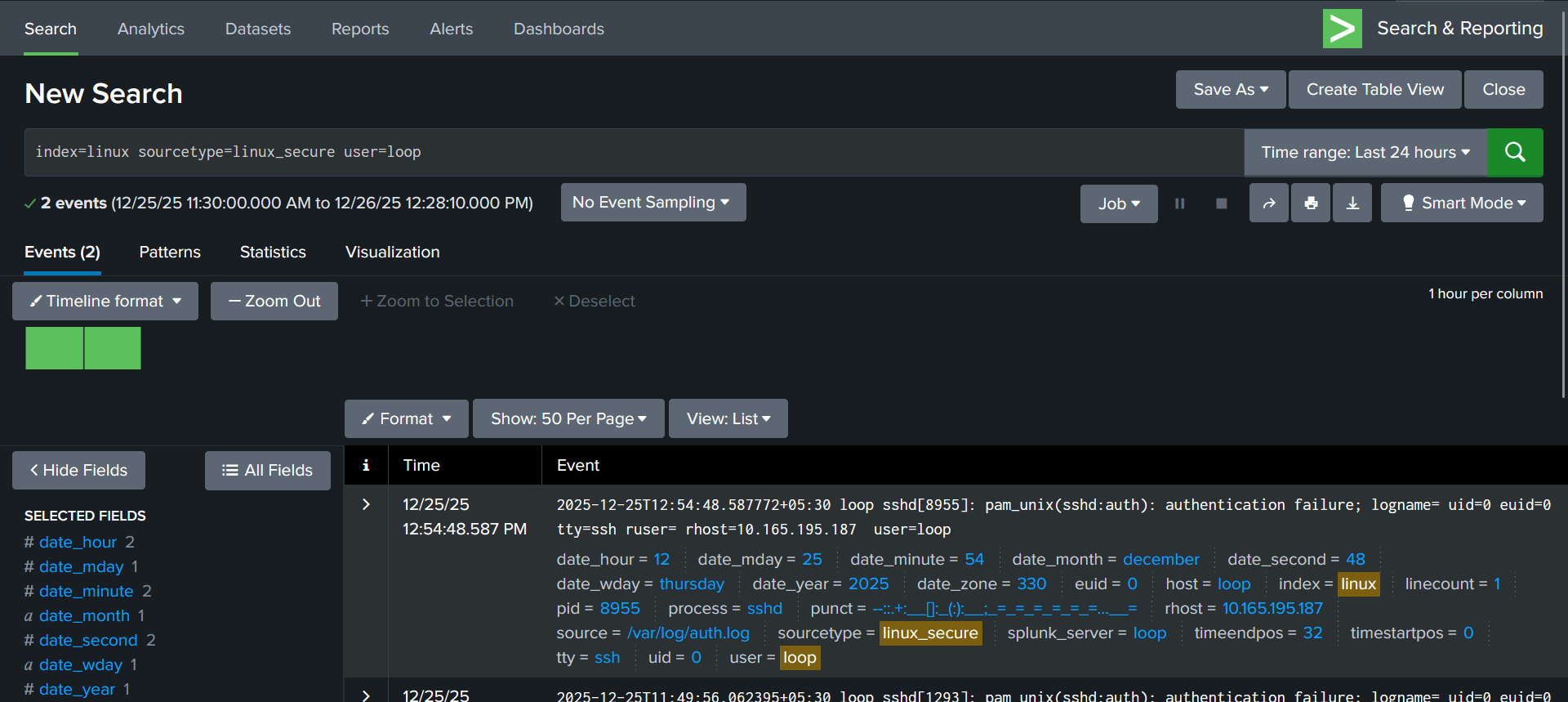
* Event count change : 2
* Time distribution (bursty vs spread) Events are **not clustered**
* Services involved (sshd, sudo) sshd
* Does this user appear normal or noisy? normal

**Ask yourself:**

“If I only knew this user’s activity, what story would I tell?”

📌 Write **one neutral observation**.

When filtered to user loop, authentication activity is very low-volume, limited to isolated SSH authentication failures, with no cron or sudo usage observed.

****

**STEP 2 — RESET TO BASELINE (CRITICAL DISCIPLINE)**

* Remove the user filter completely
* Confirm you’re back to:

**index=linux sourcetype=linux\_secure**

Never pivot directly from one filter to another.

**STEP 3 — FILTER BY A SINGLE IP (SOURCE PERSPECTIVE)**

**1️⃣ Find the IP field**

Look for:

* src, src\_ip, client\_ip, rhost

**2️⃣ Click one IP address**

**Observe:**

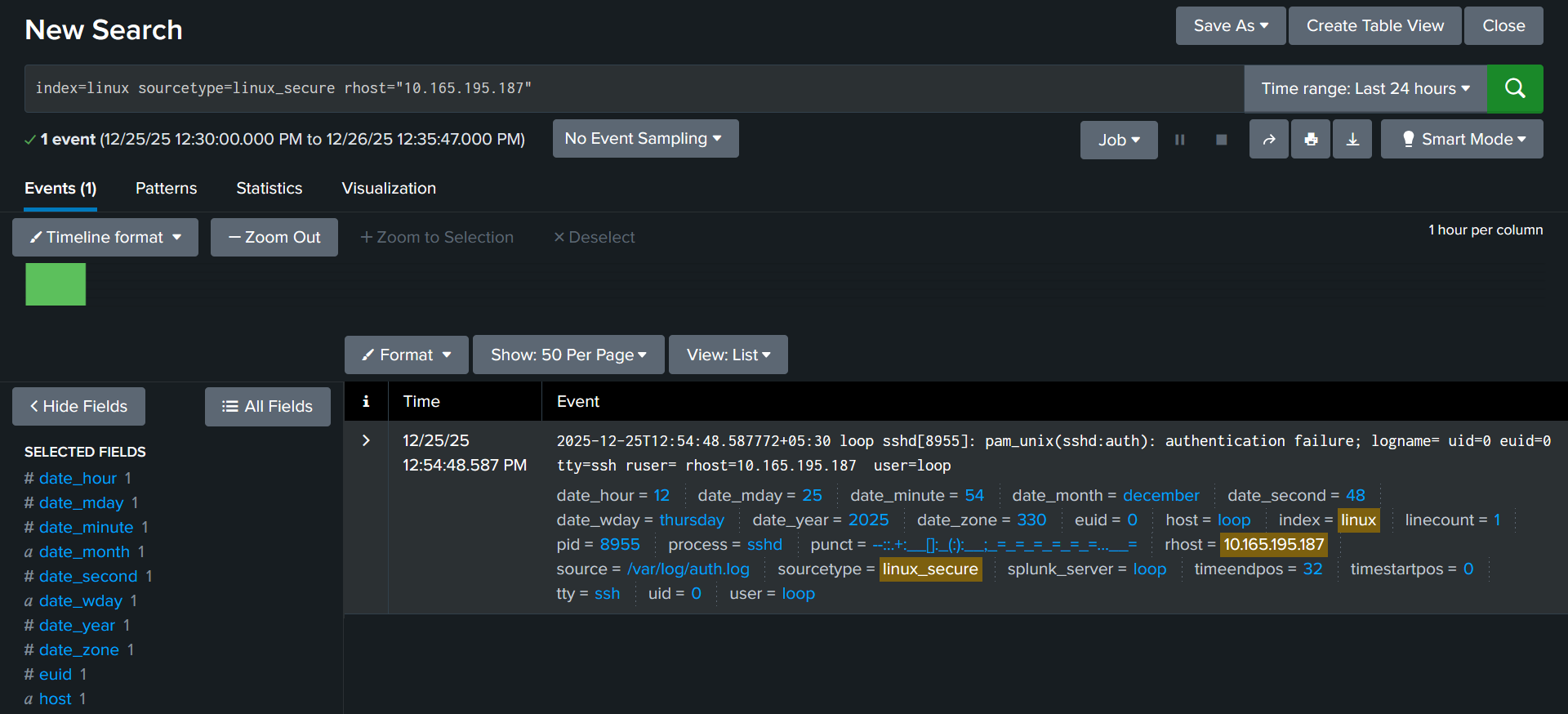
* How many users does this IP touch? ONE
* One host or many? ONE HOST : LOOP
* Tight time window or spread out? Single event

**Ask:**

“What does the activity look like *from this IP’s point of view*?”

📌 Write **one observation**.

When filtered to source IP 10.165.195.187, activity consists of a single SSH authentication failure against one user on one host, with no repeated or clustered behavior.



**STEP 4 — RESET TO BASELINE AGAIN**

Remove the IP filter.

This reset step prevents tunnel vision.

**STEP 5 — FILTER BY A SINGLE HOST (ASSET PERSPECTIVE)**

**1️⃣ Click one host value**

**Observe:**

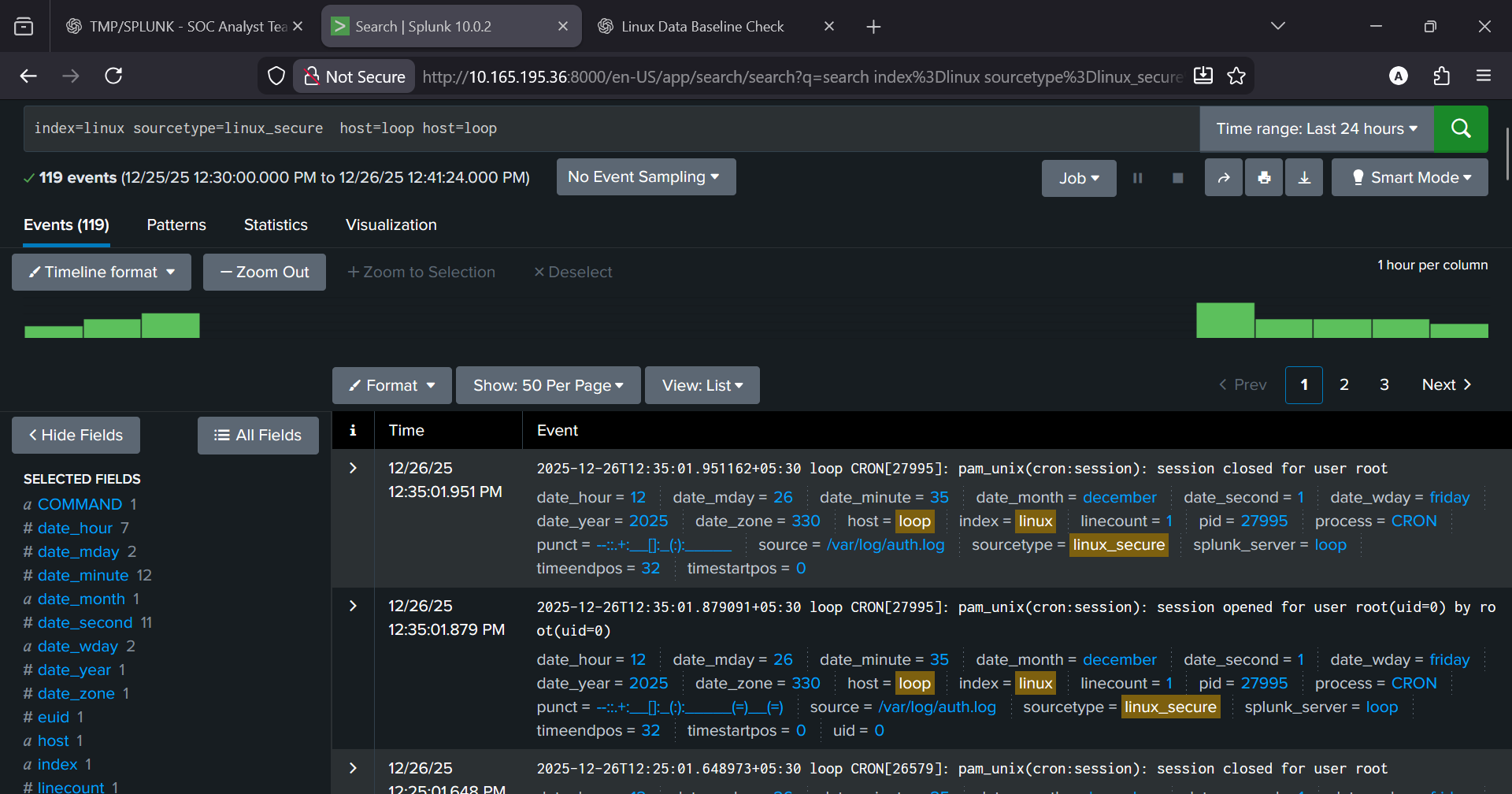
* Is this host noisier than others?
* Which users appear here?
* Which services dominate?

**Ask:**

“If this were the only host I saw, would it look unusual?”

📌 Write **one observation**.

When filtered to host loop, authentication activity is steady and predictable, dominated by routine root cron sessions with limited SSH usage and no signs of abnormal or excessive behavior.



**STEP 6 — RESET TO BASELINE AGAIN**

Yes — every time.

**STEP 7 — FILTER BY A SINGLE SERVICE (BEHAVIOR PERSPECTIVE)**

Now isolate **one behavior type**.

**Option A: SSH**

index=linux sourcetype=<auth\_sourcetype> sshd

**Option B: Sudo**

index=linux sourcetype=<auth\_sourcetype> sudo

(Keyword is OK here because service is the focus.)

**Observe:**

* Event volume
* Time pattern
* Which users appear here but not elsewhere

**Ask:**

“What does the system look like when I isolate *one behavior*?”

📌 Write **one observation**.

When isolating authentication service activity, events are moderate in volume, evenly distributed over time, dominated by cron-related root sessions, indicating normal system-driven behavior rather than interactive or malicious activity.

**STEP 8 — COMPARE THE FOUR NARRATIVES (THIS IS THE LESSON)**

You have now seen the **same data** through four lenses:

1. Baseline (everything)
2. User-focused
3. IP-focused
4. Host-focused
5. Service-focused

**Senior analyst insight:**

**Filtering doesn’t change the data.  
It changes the story you can see.**

This is why premature conclusions are dangerous.

**STEP 9 — BIAS CHECK (VERY IMPORTANT)**

Ask yourself honestly:

* Which filter made things look “most suspicious”?
* Would it still look suspicious from another lens?

If yes → worth investigating later  
If no → likely noise amplified by scope

Filtering exposes **bias**, not just behavior.

**🔥 PHASE E — INVESTIGATION TIMELINE (STORY RECONSTRUCTION)**

* Attacks are **stories over time**
* Order > frequency

**Practical**

* Reconstruct:
  + before activity
  + during activity
  + after activity
* Identify:
  + gaps
  + pauses
  + non-human timing

**🔥 PRACTICAL — INVESTIGATION TIMELINE (STORY RECONSTRUCTION)**

**🔴 STEP 0 — FIX THE VIEW (MANDATORY)**

* Time range: **Last 24 hours**
* Focus: **SSH only**

index=linux sourcetype=<linux\_auth\_sourcetype> sshd

No filters yet.

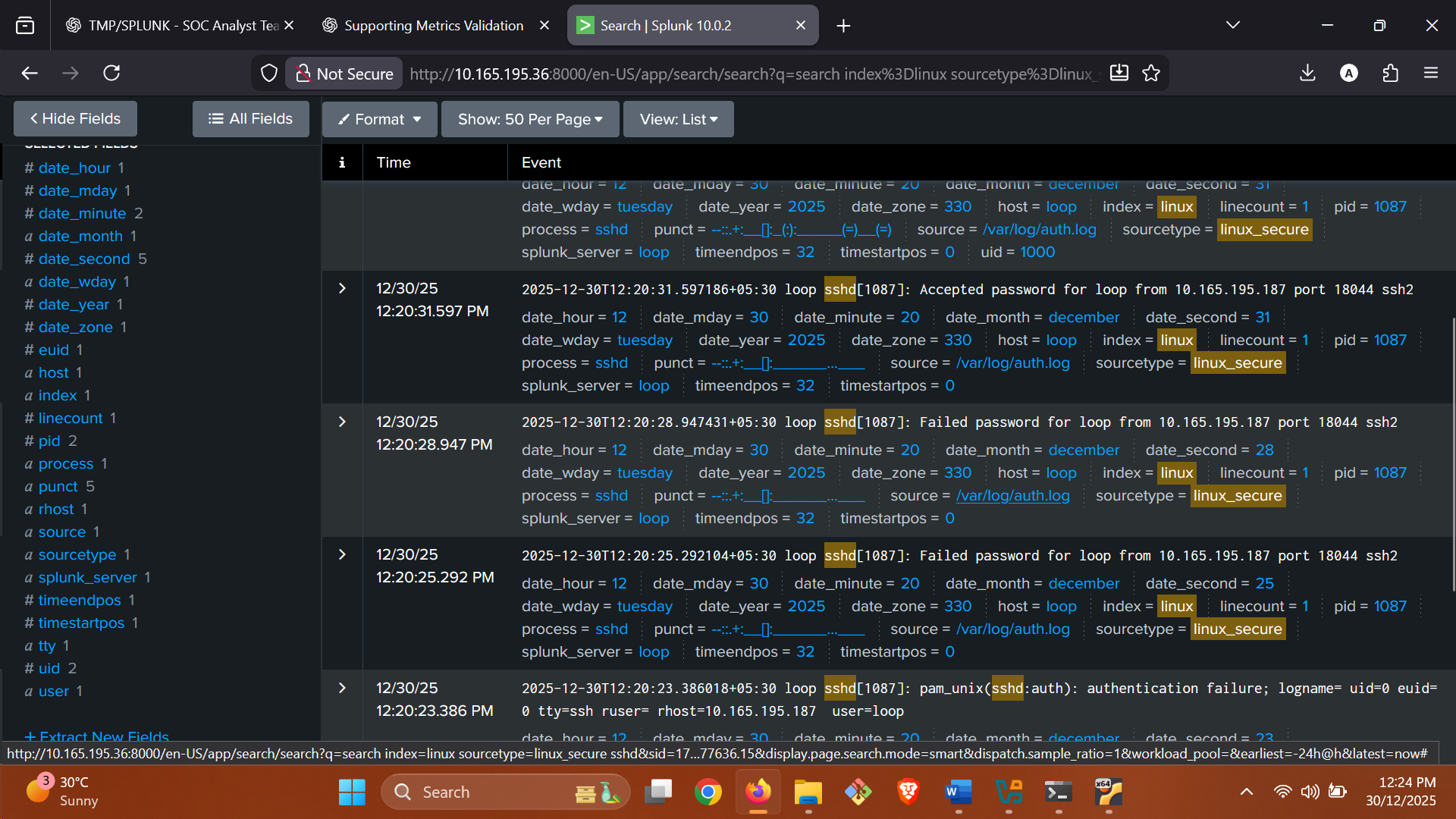
**🔴 STEP 1 — OBSERVE THE FLOW**

Just look at the events **in time order**.

Ask:

* Are events **isolated** or **clustered**? - **CLUSTERED**
* Is activity spread out or tight?

(No conclusions.)



**🔴 STEP 2 — PICK THE ANCHOR EVENT**

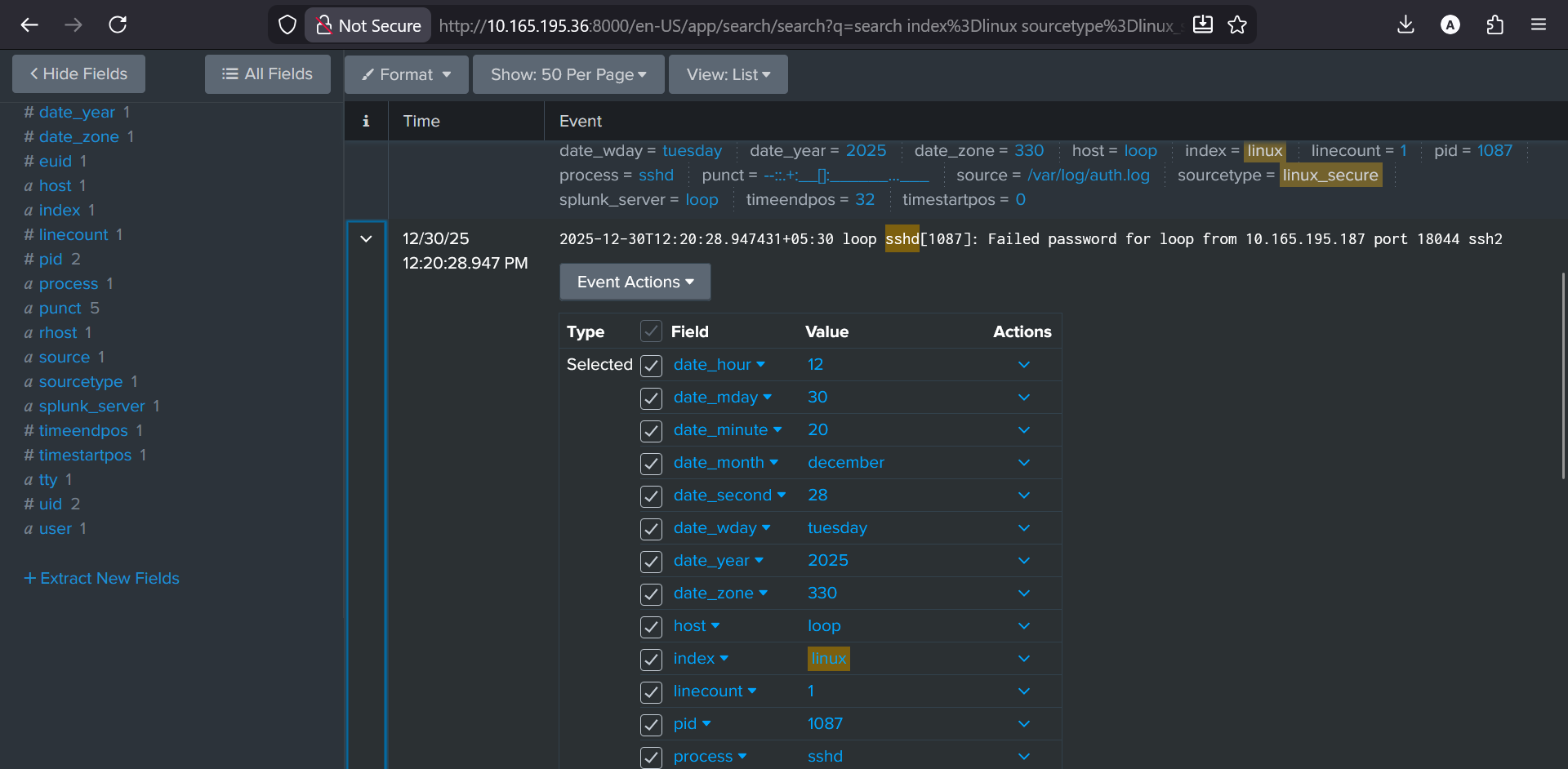
Choose **one SSH event** that triggered interest.

Note only:

* Time
* User
* Source IP

This is **NOT** the start — just the reference point.

“Anchor SSH event at 12:20:28 for user loop from source IP 10.165.195.187.”



**🔴 STEP 3 — GO BACKWARD (BEFORE)**

Scroll **up in time** from the anchor.

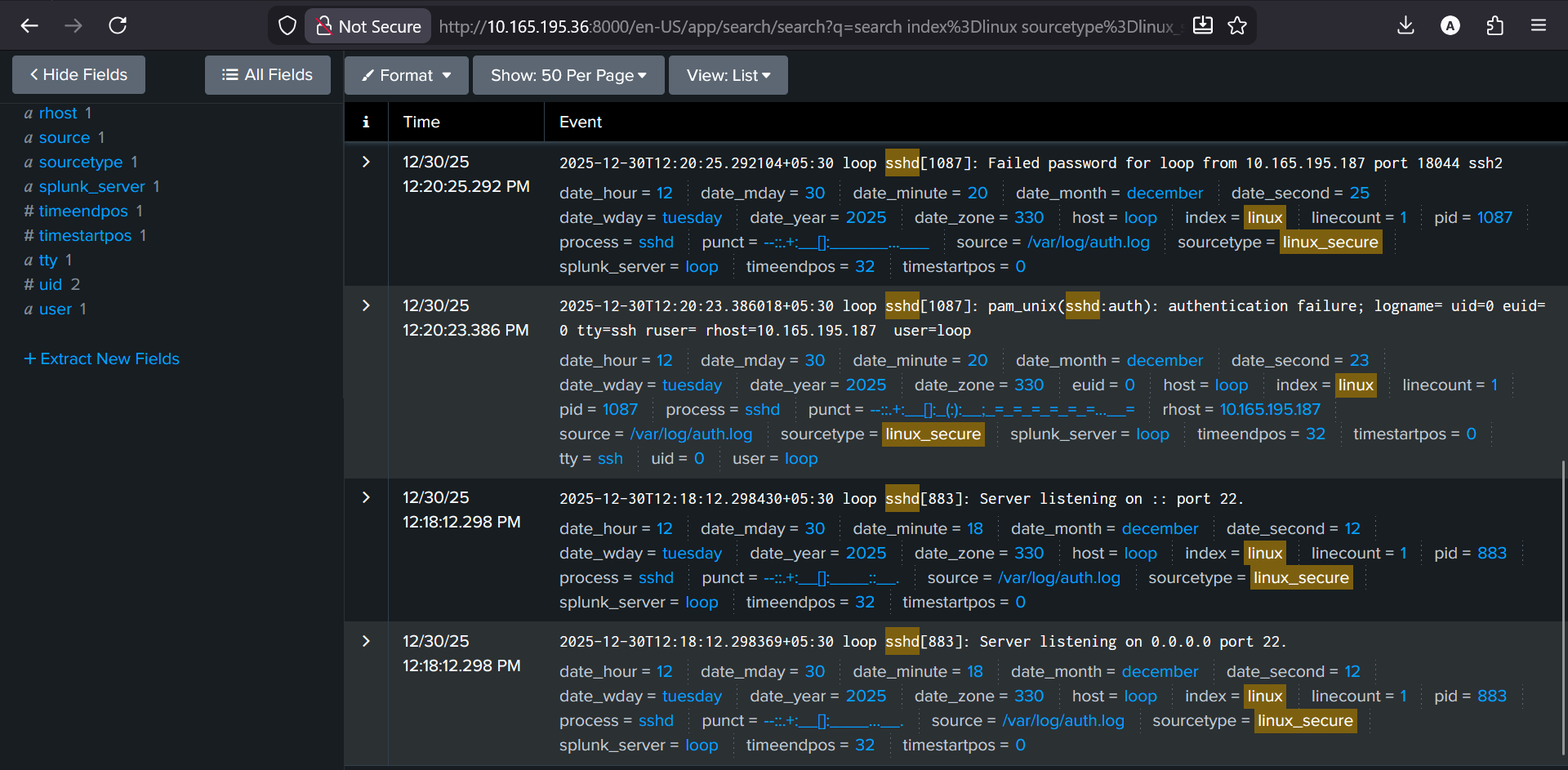
Look for:

* First time this IP/user appears
* Early failed attempts
* Slow probing

Answer:

* What is the **earliest related event**?
* **Time: 2025-12-30 12:20:23**
* **Event: pam\_unix(sshd:auth): authentication failure**
* **Actor: Source IP 10.165.195.187 targeting user loop**

This is **how the story began**.



**🔴 STEP 4 — ANALYZE THE CORE ACTIVITY (DURING)**

Focus on the cluster around the anchor.

Look for:

* Repeated failures
* Same IP/user
* Tight timing

Ask:

**Does behavior escalate?**  
**Yes.** The sequence moves from authentication failure → repeated failed passwords → successful login and session open. That’s a clear **state change**, which is escalation. Count is irrelevant; order proves it.

**Does timing feel human or scripted?**  
**Scripted.** Attempts are spaced ~2–3 seconds apart with consistent cadence. Human logins are irregular (pauses, retries, hesitation). This rhythm is mechanical.

(Order matters more than count.)

**🔴 STEP 5 — GO FORWARD (AFTER)**

Scroll **down in time** after the anchor.

Look for:

* Successful login
* sudo usage
* Session activity
* Sudden silence

Ask:

Many attacks go quiet **after success**.

**Did activity stop?**  
**Yes.** After the successful login and session open at ~12:20:31, there’s no further SSH, sudo, or session activity tied to that user/IP in the immediate window. The log goes quiet.

**Did behavior change?**  
**Yes.** Behavior shifts from rapid, repeated authentication attempts to **post-success silence**. That’s a clear change in state and intent.

**🔴 STEP 6 — IDENTIFY GAPS & PAUSES**

Read the full sequence again.

Look for:

* Long pauses
* Missing expected events
* Silence between actions

Ask:

* Logging gap or attacker waiting?**Attacker waiting (behavioral pause), not a logging gap.**

**Gaps are evidence.**

**🔴 STEP 7 — HUMAN vs AUTOMATED TIMING**

Judge **rhythm**, not volume.

* Human → irregular, pauses
* Script → fast, even, repetitive

This often matters more than “how many.

**🔴 STEP 8 — WRITE THE STORY (FINAL OUTPUT)**

**Before**

Activity began with SSH authentication failures targeting the user loop from a single source IP (10.165.195.187). The first observed action was an authentication failure, followed shortly by additional failed password attempts. This indicates initial access attempts rather than routine background noise.

**During**

The behavior escalated from repeated authentication failures to a successful SSH login. Attempts occurred in tight, consistent intervals of a few seconds, showing a mechanical cadence rather than irregular human timing. The sequence progressed in a controlled manner from failure to success without deviation, indicating deliberate access attempts rather than accidental retries.

**After**

Immediately after the successful SSH authentication and session opening, observable activity stopped. There were no subsequent SSH interactions, no sudo usage, and no visible session activity tied to the same user or source IP in the immediate window.

**Gaps / Pauses**

Expected post-login events—such as command execution indicators, privilege escalation attempts, or session closure—were absent. System logging continued for other activity, indicating this was not a logging failure. The silence represents a behavioral pause following successful access rather than missing telemetry.

**🔥 PHASE F — SUPPORTING METRICS (VALIDATION, NOT DETECTION)**

* Counts confirm patterns, they don’t define them
* Thresholds depend on baseline

**Practical**

* Count events by:
  + user
  + IP
  + host
* Compare normal vs abnormal frequency
* Tie counts back to **timeline**, not alerts

**🔥 PHASE F — SUPPORTING METRICS (STEP-BY-STEP LAB)**

**🎯 Goal of this lab**

You are **NOT detecting attacks**.  
You are **checking if the numbers agree with the story you already found**.

Story you already have:

* Same IP
* Same user
* Tight timing
* Success → silence

Now we **validate** that with counts.

**🧱 LAB SETUP (DO THIS FIRST)**

**STEP 0 — Open Splunk**

1. Go to **Search & Reporting**
2. Make sure:
   * Time range = **Last 24 hours**
   * Search mode = **Smart** (default)

Do **NOT** change anything else.

**🔹 PART 1 — COUNT BY USER**

**What this means**

“Which users appear most in SSH logs?”

**STEP 1.1 — Run the base search**

In the search bar, type **exactly** this:

index=linux sshd

Press **Search**.

✅ This shows **all SSH activity**, nothing filtered.

**STEP 1.2 — View user counts (NO SPL)**

Look at the **left sidebar**.

1. Find **user**
2. Click on it

Splunk now shows:

* user name ; loop
* how many events each user has 7



**STEP 1.3 — What to observe**

Ask yourself:📌 You are NOT judging yet.

**Does user loop appear?**  
**Yes.** The user field shows loop.

**Does loop have activity only once or in a small burst?**  
**Only once** in the current result set (1 event with user=loop, 14.286% of total events because only 1 of 7 events contains a user field).

You are only **observing distribution**.

**🔹 PART 2 — COUNT BY IP**

**What this means**

“Which IPs are involved in SSH activity?”

**STEP 2.1 — Same search, new view**

Do **NOT** change the search.

Still:

index=linux sshd

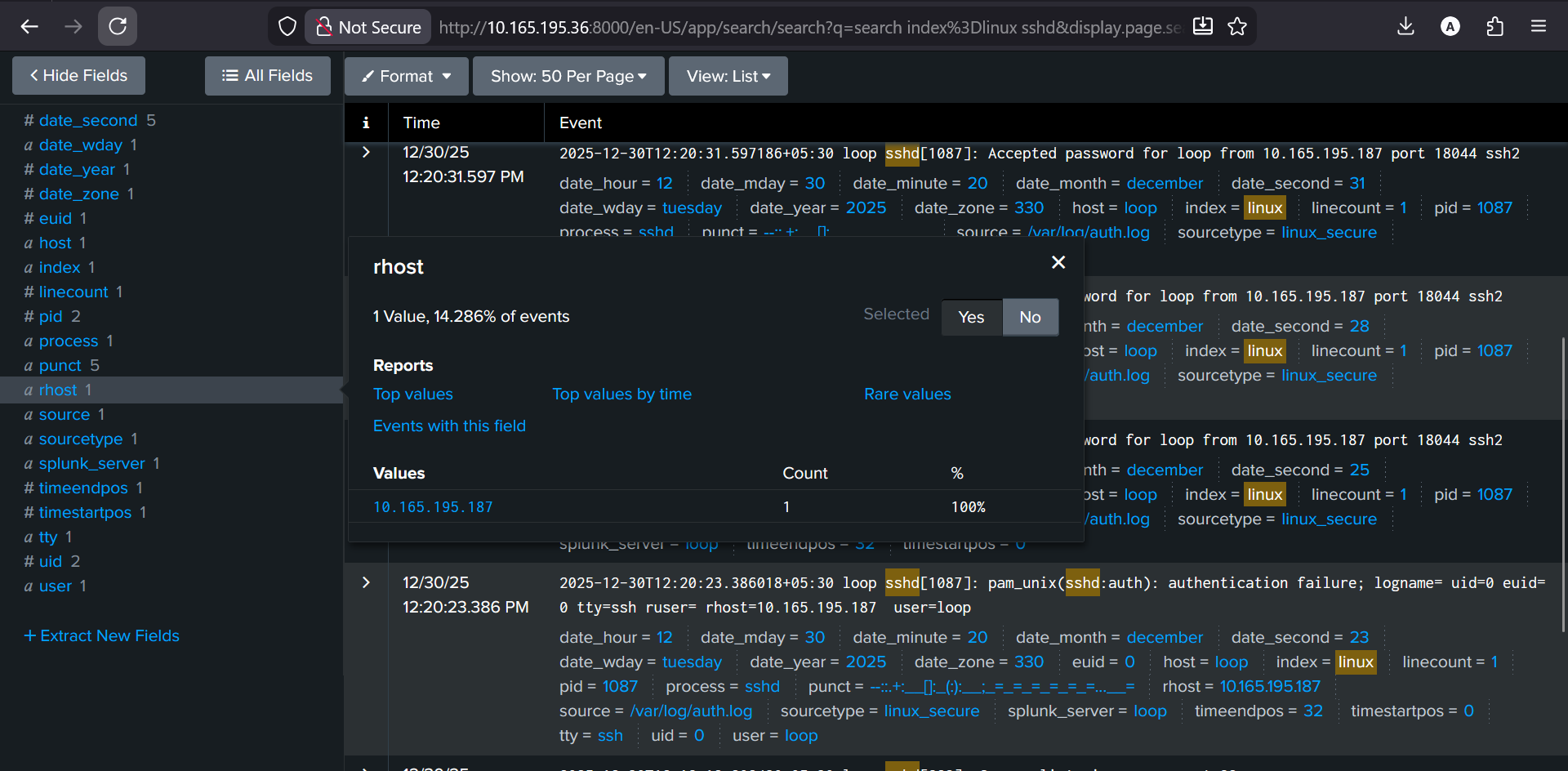
**STEP 2.2 — Click IP field**

In the left sidebar:

1. Find **src\_ip** or **rhost**
2. Click it

Now you see:

* IP addresses
* Event counts per IP



**STEP 2.3 — What to observe**

Look for:

* **Does IP 10.165.195.187 appear?**  
  **Yes.**
* **Does it appear only a few times?**  
  **Yes.** It appears **once** in the events that contain the rhost field.
* **Are those events close together in time?**  
  **Yes.** The IP appears during the same short SSH activity window you identified earlier, not spread across hours.

📌 Few events + tight timing = important  
📌 Many events spread out = often normal

**🔹 PART 3 — COUNT BY HOST**

**What this means**

“Is one machine being targeted?”

**STEP 3.1 — Same search again**

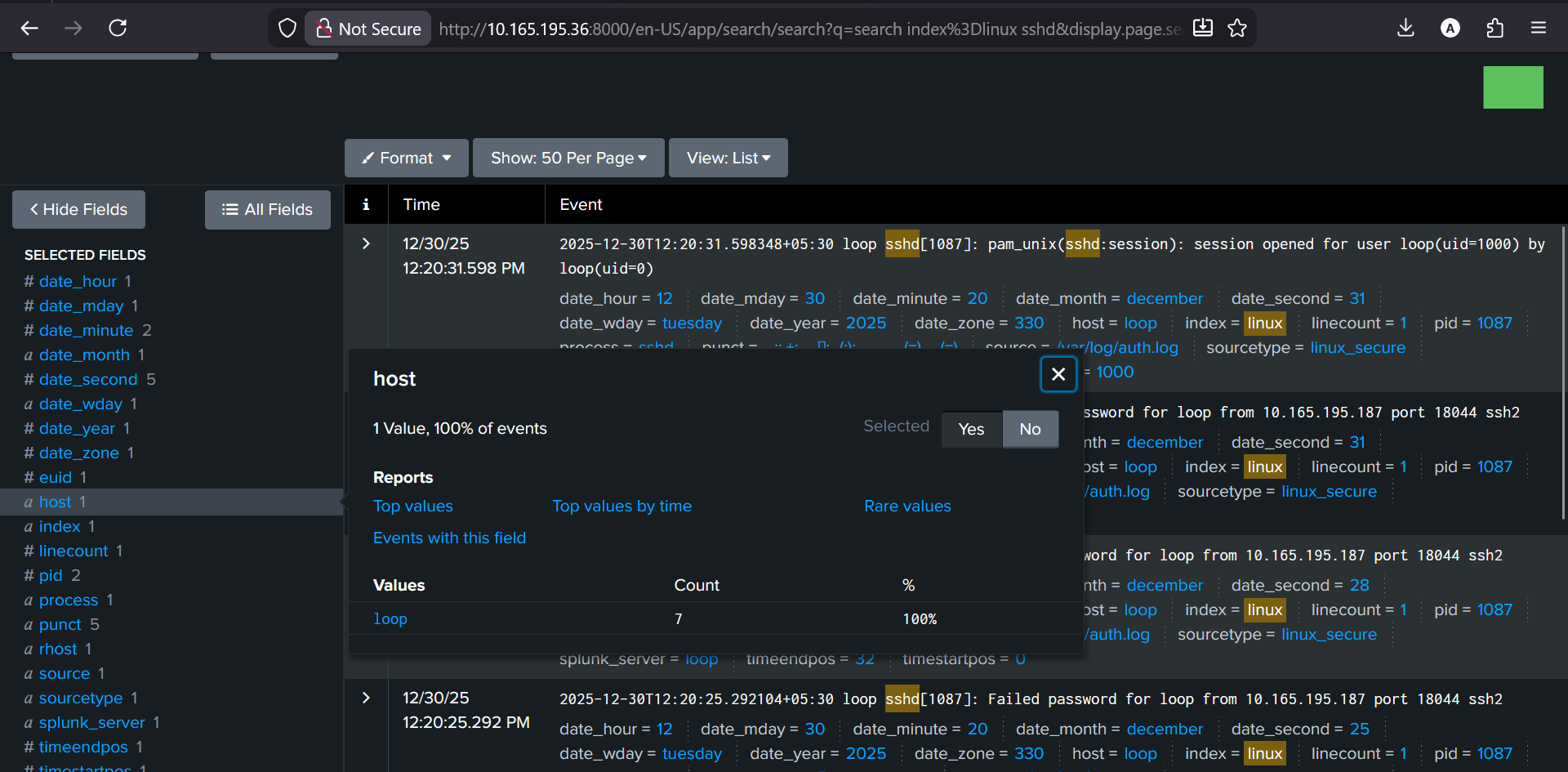
Still:

index=linux sshd

**STEP 3.2 — Click host**

Left sidebar:

1. Find **host**
2. Click it



**STEP 3.3 — What to observe**

Ask:

* Is only **one host** involved? Yes
* Or are multiple hosts equally noisy? No

📌 One host → targeted behavior  
📌 Many hosts → background activity

**🔹 PART 4 — NORMAL vs ABNORMAL (VERY SIMPLE)**

**This part scares beginners — don’t overthink it.**

**STEP 4.1 — See “normal”**

1. Set time picker to **Last 24 hours**
2. Look at SSH activity:
   * scattered?
   * low and steady?

This is your **normal baseline**.

**STEP 4.2 — See “abnormal”**

1. Change time picker to **Last 5 minutes**
2. Look again

Ask ONE question:

“Does this look different from before?”

If yes → behavior changed  
If no → likely noise

**🔹 PART 5 — CONNECT COUNTS TO YOUR STORY (MOST IMPORTANT)**

Now say this **in words** (not SPL):

* Same **user** shows activity in a short burst
* Same **IP** appears briefly, not constantly
* Same **host** is involved
* Activity **stops after success**

👉 That means:  
**Counts support the timeline.**

If counts didn’t match, you would **re-check the story**.