Detection Coverage Analysis

Complete Step-by-Step Guide

Identify Coverage Gaps & Dormant Rules

Generated: January 15, 2026

Google SecOps / SIEM Detection Analysis

Table of Contents

Overall Purpose

This process identifies **detection coverage gaps** in your security environment by comparing:

- **Raw events** ingested into your system
- **Detections** triggered by your security rules

The goal is to answer critical questions:

- *"What events are we collecting but NOT detecting?"*
- *"Which enabled rules are dormant and potentially broken?"*
- *"Where do we have blind spots in detection coverage?"*

■■ CRITICAL LIMITATION - Read This First

What This Analysis DOES Show:

■ Events occurring in your environment but NOT detected by any rule

■ Detection coverage gaps for actual occurring events

■ Rules actively triggering and detecting threats

What This Analysis DOES NOT Show:

■ Enabled rules that have NO recent detections (dormant rules)

■ Rules that exist but never triggered in the query time window

■ Rules that are broken or misconfigured (silently failing)

The Critical Problem:

If an enabled rule never triggers, it won't appear in the detection results. This means you could have:

• Rules that are **broken or misconfigured** (silently failing with no alerts)

• Rules that are **enabled but never matching** (overly restrictive conditions)

• Rules that **aren't working as intended** (detection dependencies not met)

• **Security blindspots** for threats you think you're monitoring

Real-World Example: Failed Login Rule (4625)

Your environment is generating **failed login events (4625)** in high volume, but these rules are **enabled yet showing ZERO triggers:**

• WIN_Excessive_Account_Lockout_Multiple_Users → ENABLED but dormant

• WIN_Excessive_Account_Lockout_Same_User → ENABLED but dormant

Questions this should raise:

■ Why aren't these rules triggering despite 4625 events occurring?

■ Is the rule condition too restrictive?

■ Are the log sources correctly mapped in the rule?

■ Is the rule actually enabled in the production environment?

■ Are detection dependencies (like UDM enrichment) being met?

■ Has the rule logic been accidentally disabled or modified?

This is a RED FLAG that requires immediate investigation.

Step 1: Identify Top Log Volume Sources

Purpose:

Understand your data landscape before analyzing detections

Function:

Quantify how much data each log type generates weekly to identify high-volume sources and understand scale

Why it matters:

- High-volume log types have more detection opportunities and potential gaps
- Low-volume types may be security-critical but easy to miss
- Helps prioritize which log types need better coverage

Output:

- Log type (e.g., winevtlog, fortinet_firewall, office_365, powershell)
- Weekly event count per log type
- Weekly data volume in GB

Action:

Reference only (optional). Helps context for Steps 2-3.

Step 2: Extract Rules and Their Detection Activity

Purpose:

Map what your detection rules are actually catching

Function:

List all detection rules that have triggered recently and show how many times each rule has triggered, what threats/behaviors the rule detects, rule severity/category/authorship, and when the rule last triggered

■■ IMPORTANT: This query must be run in a Dashboard, NOT in a native query interface.

Steps:

1. Go to your Google SecOps **Dashboard** (not native query)
2. Create a new dashboard table or use existing table
3. Paste the query into the dashboard query editor
4. Run the query in the dashboard
5. Export results as CSV
6. **Save as detection_results.csv**

Output:

- Rule name
- Number of alerts triggered (in query period)
- Rule description and summary
- Threat/behavior category
- Rule severity level
- Last trigger timestamp
- Estimated data volume per day
- **Rules listed are ONLY those that have triggered**

Step 3: Capture All Raw Event Data

Purpose: Get a complete picture of all events occurring in your environment (regardless of whether they're detected)

Steps: Run query → Export as **event_results.csv**

Step 4: Run Python Comparison Script

Purpose: Identify which events are NOT being detected

Inputs: detection_results.csv + event_results.csv

```
Command: python compare_unmapped_events.py
```

Outputs: Console summary + HTML report (unmapped_events_report.html)

Key Takeaways

The Three Critical Findings:

1. Unmapped Events (from script)

- Events occurring in your environment
- NOT detected by any rule
- May indicate missing rules or misconfigured rules
- High-risk ones need investigation/remediation

2. Dormant Rules (manual verification)

- Rules that are ENABLED
- Show 0 detections in query period
- Strong indicator of rule misconfiguration/breakage
- **MUST be investigated - they represent blind spots**

3. Active Rules (from script)

- Rules actively triggering
- Providing detection coverage
- Baseline for comparison

Critical Success Factor:

You MUST perform Step 6 (manual verification) to find dormant rules. The Python script alone is incomplete. An enabled rule with zero detections despite matching events is a RED FLAG requiring immediate investigation.

Checklist for Execution

Before You Start:

- ■ Access to Google SecOps / SIEM query interface
- ■ Export permissions for CSV
- ■ Python 3.6+ installed locally
- ■ Text editor for reviewing CSVs
- ■ Access to Rule Management UI for Step 6

Step 2 - Detection Rules:

- ■ Query runs in Dashboard (not native query)
- ■ Results show rule names and trigger counts
- ■ Export to CSV as detection_results.csv
- ■ File contains 50+ rule names (verify not empty)

Step 3 - Raw Events:

- ■ Query runs successfully
- ■ Results show log_type, event_type, product_event_type
- ■ Export to CSV as event_results.csv
- ■ File contains 100+ combinations (verify not empty)

Step 4 - Python Script:

- ■ Script runs without errors
- ■ Browser opens with HTML report
- ■ Report shows summary statistics
- ■ Coverage percentage displayed

Step 6 - Manual Verification:

- ■ Export enabled rules from Rule Management UI
- ■ Compare against detection_results.csv
- ■ Identify dormant rules (enabled but 0 triggers)
- ■ Investigate each dormant rule
- ■ Document root causes and actions

Glossary

Detection: A security event that triggered an alert from an enabled rule

Unmapped Event: An event occurring in your logs but not detected by any rule

Dormant Rule: A rule that is enabled but showing zero triggers (not detecting anything)

Coverage: Percentage of events being detected by rules (mapped / total)

Event Type: Behavioral category (USER_LOGIN, PROCESS_LAUNCH, etc.)

Product Event Type: Specific event identifier (4625, 4688, traffic-forward, etc.)

Log Type: Source system (winevtlog, fortinet_firewall, office_365, etc.)

UDM: Unified Detection Model - normalized event format in Google SecOps

Document Information

Version: 1.0

Generated: January 15, 2026 at 07:27 PM

Maintainer: Security Operations Team

This guide provides comprehensive detection coverage analysis for identifying security blind spots and dormant detection rules in your environment.