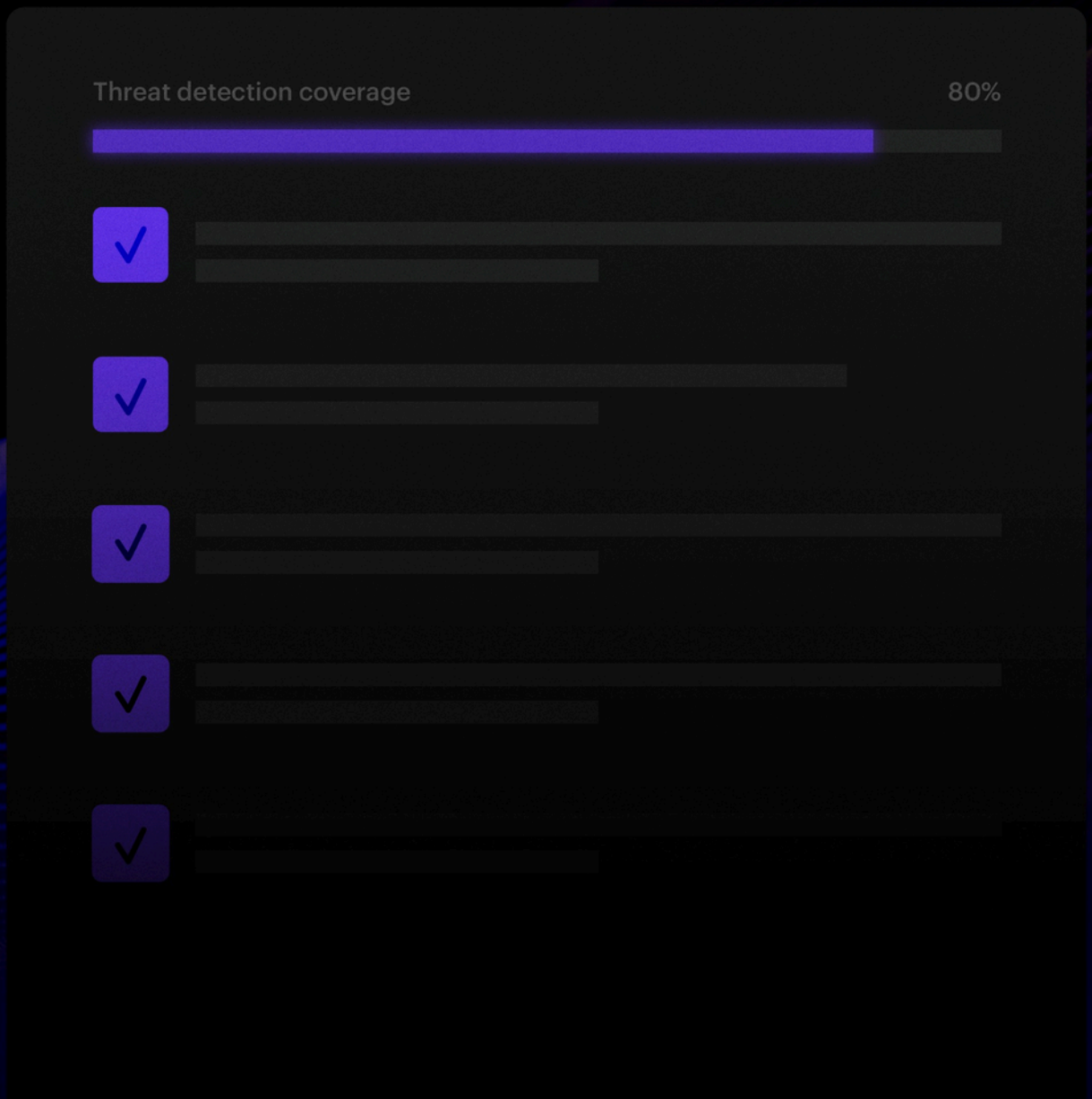
HUNTERS

THE ULTIMATE DETECTION ENGINEERING CHECKLIST

A 10-Step checklist for SOC teams



10 Step Detection Engineering Checklist

Practical Advice Based on Field Experience

Authors: Ophir Kelman & Yagel Yosef, Security Researchers at Hunters

1. Start With Strategy: Define Purpose and Scope

Define the Detection Goal

Understand the reason for building the detection:

- Is it to expand existing attack surface coverage?
- Is it addressing a newly discovered threat?
- Are you trying to monitor for a tactic seen in recent threat intelligence?

Key Learning No. 1
Be mindful of the naive approach of checking every box in the MITRE matrix. Instead, each detection should be rooted in a clear and justified need. Without this clarity, the rule may not align with actual threats or gaps in coverage.

Clarify Data Requirements

- Identify what logs are required for **detection**
- Also identify what logs are necessary for **investigation** (triage and deep dive)
- Understand how your platform supports log access and query design

Simulate Early

Run simulations during creation:

- Validate what the rule detects and what it misses

Use this as a form of lightweight threat hunting to assess "noise versus signal"

Key Learning No. 2
A detection relying on whitelist maintenance will create a heavy burden on SOC analysts. If analysts must frequently update rule exclusions due to noise, the rule becomes a liability rather than an asset.

2. Build With the Right Tools and Technology

Align With Platform Capabilities

Determine whether your tech stack supports:

- Joins
- Time-series analysis
- Machine learning models

Caution:
Some platforms don't support joins, and others cannot apply advanced analytics. Know your tooling limits upfront so you don't build detections that can't execute properly.

Avoid Over-Reliance on Exclusions

Rules with complex `[NOT A AND NOT B AND NOT C]` logic tend to accumulate noise and become unsustainable.

Key Learning No. 3
A rule built with heavy exclusion logic can become unmaintainable, leading to the decision to discard it entirely rather than chase noise indefinitely.

3. Prioritize Context-Rich Alerts

Provide Rule-Specific Context

- Example: "This user typically doesn't request TGS with weak encryption"
- Helps analysts focus by highlighting anomalies within familiar baselines

Enrich With General Threat Intelligence

- IP reputation, geolocation, ASN, and threat feeds add essential color to alerts

Ensure Context is Easily Accessible

- Avoid requiring analysts to pivot across multiple tools
- Embed context in the alert dashboard or ticket if possible

Key Learning No. 4
Analysts often fall into investigative “muscle memory,” skipping steps or dismissing real threats because the process becomes routine. Enriching alerts with high quality context disrupts that autopilot behavior and brings critical alerts into sharper focus.

4. Measure What Matters — KPIs

Signal-to-Noise Ratio (SNR)

- Chronic noise = constant across environments
- Acute noise = caused by localized or transient events

Key Learning No. 5
Security Researchers Ophir Kelman & Yagel Yosef noticed a detection became noisy only after a new service account was deployed. This shows the importance of monitoring for both chronic and environment-specific noise patterns.

Silent Rules

Monitor for rules that never fire:

- Could be due to changes in log structure or logic errors
- Might also be valid detections for rare events

Caution:

Silent rules often go unnoticed because most KPIs focus on noisy alerts. However, detection systems can silently fail if vendors alter log formats unexpectedly.

False Positives vs. Benign True Positives

- False positives: Alert fires incorrectly
- Benign TPs: Alert fires correctly, but activity is non-malicious

Anecdote:
A rule detected Mimikatz execution based on process name. If a file named `mimikatz.docx` triggered the rule, that would be a false positive. If a pentester actually ran Mimikatz during testing, it's a benign TP. The challenge is in choosing whether to exclude or retain such cases.

Key Learning No. 6
Blindly excluding benign true positives (e.g., sysadmins doing weird things) can blind you to attacker activity that mimics the same behavior.

5. Differentiate SOC Rules vs. Threat Hunting Rules

SOC Rules

- Should be low-noise, high-confidence
- Designed for continuous monitoring and immediate triage

Threat Hunting Rules

- Can tolerate higher noise
- Allow analysts to proactively explore weaker signals

Key Learning No. 7
Security Researchers Ophir Kelman & Yagel Yosef discovered a rule detecting internal network scanning was too noisy for SOC Queue/Alert Queue inclusion but proved effective in threat hunting. Treat such rules as auxiliary tools, not production-grade alerts.

6. Use Threat Prevalence to Guide Prioritization

Evaluate Prevalence in the Wild

- Use Microsoft's telemetry data if available
- Use quarterly threat reports from CrowdStrike, Check Point, Microsoft, etc.
- Consider LLM tools (e.g., NotebookLM) to process and extract insight from multiple reports

Key Learning No. 8
Security Researchers Ophir Kelman & Yagel Yosef questioned a detection due to noise. Only after verifying that the tactic it targeted was widely used in the wild did the team decide to continue supporting and refining the rule.

7. Leverage Analyst Feedback and Engagement

Use the Case Management System

- Export logs and look for high-frequency keywords like "false positive"
- Identify detections with overly complex investigation flows

Key Learning No. 9
An internal tool flagged the most "clicked" alerts (engagement) in the SOC. By repeatedly refining these high-impact rules, the team in question became so efficient that they had to design a new monitoring approach.

Detect Engagement Trends

- Frequent dismissals suggest poor alert design
- Rich context may improve alert stickiness and investigation depth

8. Commit to the Detection Engineering Lifecycle (DELC)

Continuously Monitor and Improve

- Maintain alerting for:
    - Chronic and acute noise
    - Rule silence
    - High engagement but poor quality
- Automate identification of rules that need tuning

Key Learning No. 10
A small SOC team working at a large organization created KPIs on their KPIs — ‘meta-monitoring’ to validate that their SNR (signal-to-noise ratio) and engagement metrics were still providing value over time.

Incorporate External Testing

- Red team simulations
- Penetration tests
- User satisfaction metrics and feedback loops

9. Optimize for Performance and Cost

Query Efficiency Matters

- Avoid unnecessary 90-day lookbacks. Start small (e.g., 7 days), and expand only if needed
- Inefficient queries delay alerts and strain resources

Key Learning No. 11
Slow-performing rules delayed alert generation. By the time SOC analysts working at a medium sized organization saw them, the threat had already progressed. Tightening the query window improved both performance and detection timing.

Cost Monitoring

- Heavy or frequent detections on cloud platforms can incur unexpected processing charges

10. Manage Third-Party vs. Internal Rules

Don’t Treat All Alerts Equally

- Vendor-supplied rules may be noisy and offer little insight
- Use vendor confidence scores to filter:
    - Drop low-confidence alerts (e.g., below “medium”)

Key Learning No. 12
Third-party rules often fire excessively and lack flexibility. Setting confidence-based thresholds helped reduce clutter and preserved analyst focus.

Final Thoughts

Detection engineering is not static. It is a **cyclical process** of innovation, validation, measurement, and feedback. Teams that regularly review, refine, and revisit their rules will stay resilient against evolving threats. Hunters is an AI-Powered, Next-Gen SIEM for Small teams that want a sophisticated and streamlined approach to security operations. Using Hunters, SOC teams significantly enhance analyst efficiency and threat response accuracy. To find out how Hunters can help your lean SOC team, request a demo.

Connect with the authors: Yagel Yosef & Ophir Kelman.