

What to Monitor

Pattern: deciding what is worth watching. This is the judgment page, and it is deliberately general. It walks the categories worth monitoring and the reasoning for each, without the thresholds or exact detection logic, because those are the part that helps an attacker if published. The transferable value is the reasoning: *why* each category earns a place, and what class of trouble it catches.

The governing principle is that you cannot watch everything, and trying to would bury the signal that matters under the volume that does not. So monitoring is a series of deliberate choices. Watch the categories most likely to reveal trouble, and accept the risk of what you are not watching. Here are the categories that earn their place, and why.

Authentication events

Why it matters: authentication is where an outsider becomes an insider, so it is the single most valuable thing to watch. A burst of failed logins, attempts against accounts that do not exist, or a sudden change in the pattern of who is authenticating from where, are the earliest visible signs of someone trying to get in.

What it catches: brute-force attempts, credential-stuffing, username enumeration (probing to discover which accounts exist), and the difference between a real user fumbling their password and an automated tool grinding through guesses. The identity provider and any service handling logins are the sources.

The reasoning: because centralized authentication funnels most logins through one system, watching that one system's events gives broad coverage cheaply. This is a direct dividend of consolidating identity; one place to watch for the most important category of event.

Boundary and perimeter events

Why it matters: these are the events from the controls that sit at the edges; the firewall, the intrusion-prevention tooling, the web application firewall, the VPN. They record what was refused at the boundary, which is a running picture of what is probing you.

What it catches: port scans and connection attempts the firewall dropped, IPs that tripped automated banning, requests the web application firewall refused, and connection attempts to the VPN. Individually these are routine background noise; in aggregate, a spike or a pattern is the

signal. A sudden surge of drops from one source or a coordinated probe across services.

The reasoning: the boundary controls are already making decisions and logging them. Watching their output turns controls that silently defend into controls you can see defending, which is both reassuring when quiet and informative when not.

System health

Why it matters: not every threat is an attacker. A host that runs out of disk, exhausts memory, or pegs its CPU can take a service down as surely as an intrusion, and a resource anomaly can also be a *symptom* of one (a sudden disk fill from logging a flood of attacks, a CPU spike from something that should not be running).

What it catches: hosts approaching resource exhaustion, unusual load, and disk filling over time. This is the category with the most operational (rather than strictly security) value.

The reasoning: availability is part of security, and health monitoring is cheap. The same pipeline that watches for attackers may as well watch for the mundane failures that are statistically far more likely to actually take something down.

Critical-severity events

Why it matters: the logging system already classifies events by severity, and the highest levels, the errors and critical conditions, are worth surfacing regardless of which service produced them. A critical event is by definition something the system itself considered serious.

What it catches: service failures, crashes, and the kind of severe conditions that a service flags on its own. Rather than needing a bespoke check per service, a single rule that surfaces anything above a severity line catches the serious problems across everything at once.

The reasoning: this is high-value and low-effort. The severity classification is free and already applied; watching the top of it is a broad net for "something is seriously wrong" that needs no per-service tuning.

Service-specific signals

Why it matters: a few services warrant their own attention beyond the generic categories, either because they are especially sensitive (a password manager, the identity provider) or because they emit signals worth watching that the generic categories would not capture (new account registrations, specific security-relevant actions).

What it catches: things particular to a service, such as a new user registering somewhere that should rarely see new users, a sensitive administrative action, a service-specific error pattern. These are narrow, deliberate additions on top of the broad categories.

The reasoning: the generic categories give broad coverage; a small number of service-specific checks add depth exactly where the environment's most sensitive services are. Add these sparingly, for the services that genuinely warrant it, not as a reflex for every service.

The coverage-versus-noise tradeoff

Every category above is a choice to watch something, and implicit in each is a choice not to watch a hundred other things. The goal is not total coverage, it is the highest-value coverage that stays quiet enough to be trusted. A monitoring system that watches everything generates so much noise that its alerts get ignored, which is worse than watching less and actually reading what it reports. Choose the categories that reveal the most trouble per unit of noise, and be disciplined about not adding checks that generate more false alarms than genuine findings.

Where exactly the lines sit within each category, the thresholds, the windows, the specific conditions, is the tuning that this book deliberately does not publish. Those values are where an attacker would look to learn what they can do without tripping an alert, and they are specific to an environment's normal traffic in any case. The categories and the reasoning transfer; the tripwire settings you work out for yourself, against your own baseline.

Adapt this for...

Any environment deciding what to monitor. The categories, authentication, boundary, health, critical severity, and a few service-specific signals, are a sound starting set for almost any small environment, because they concentrate on where trouble first becomes visible. Adapt the specifics to what you run, keep the discipline of watching high-value categories rather than everything, and set your own thresholds against your own normal. The judgment, not the numbers, is the thing worth carrying over.

Revision #2

Created 2026-07-31 20:13:49 UTC by Chris Landis

Updated 2026-07-31 20:29:24 UTC by Chris Landis