zsoftly

WHITEPAPER

SOC 2 Readiness and Compliance Evidence with Wazuh

How we use an open source SIEM for security evidence: twelve dashboards, eleven in the monthly report pack, protected object versions, retained logs, and lessons from building the pipeline.

AUTHOR
Ditah Kumbong, CTO and Founder of ZSoftly

DATE
November 2026

VERSION
1.0

AUDIENCE
Security, compliance and platform teams

CONTENTS

Executive summary	Page 2
Why a Type 2 report needs continuous evidence	Page 3
Mapping Wazuh to the Trust Services Criteria	Page 4
The twelve dashboards, as they run today	Page 5
The monthly evidence pack and retention	Page 7
Signals beyond the agent	Page 8
Lessons from production and a readiness checklist	Page 9
Where Wazuh stops, and how ZSoftly can help	Page 10
Sources	Page 11

Standards and vendor facts in this paper were checked against their published documentation on 15 November 2026. The design, screenshots and lessons come from the security monitoring that ZSoftly runs for its own cloud platform and staff devices. Identifying values and headline totals are masked in the screenshots. Some chart counts remain visible.

EXECUTIVE SUMMARY

Collect the evidence every month, not the week before the audit

A SOC 2 Type 2 report asks whether your controls worked across a whole observation period. That makes evidence a running cost. We use Wazuh, an open source security platform, to collect security monitoring evidence. Twelve dashboards cover access, vulnerabilities, configuration, file integrity, malware, threat triage, agent coverage, change, attack origins and leaked secrets. Each one opens with a short "How to read this" panel so a reviewer knows what the numbers mean. A monthly job renders 11 of the 12 dashboards to PDF, hashes them and stores protected versions under a seven-year S3 compliance lock. Secrets Exposure is not yet in the scheduled pack. Operations and security receive the reports by email. Alert data stays searchable for 12 months and sits in locked cold storage for 18 more, the split set out in US federal memo M-21-31. We restore from cold storage every quarter to prove the copies are usable. This paper shows how the pieces fit and what went wrong on the way.

The problem

- Evidence lives in screenshots taken by hand, often after the fact.
- Log retention is set by disk size, so data from early in the window is gone when the auditor asks.
- Raw security numbers alarm people. Thousands of findings look like a crisis without context.
- Nobody notices when a scheduled check stops running.

The approach

Put every server and staff device under one Wazuh deployment. Keep the dashboards in Git and map each one to the criteria it supports. Render a monthly evidence pack automatically and protect its object versions from changes through the S3 API. Privileged storage administration remains a separate control. Feed extra signals into the same platform: CrowdSec bans, anomaly detection and secret scanning. Review the pack every month and treat gaps as findings.

One source
Servers and staff devices report to one platform.

Twelve dashboards
Each mapped to SOC 2 criteria and stored in Git.

Locked monthly
PDFs with SHA-256 hashes under a 7-year lock.

30 months of logs
12 searchable, then 18 in locked cold storage.

Who should read this

Compliance and security leads should read this page, the mapping on page 4 and the evidence pack on page 7. **Platform engineers** will find the build details on pages 3, 7 and 8.

Anyone starting a SOC 2 program should work through the lessons and checklist on page 9 and the governance layer on page 10 before they buy a tool.

> **What this paper is not.** It is not audit advice and it does not describe a completed SOC 2 examination. Dashboards are evidence. An auditor still decides whether a control is designed well and operated across the period. Wazuh's own documentation notes that its control mappings are indicative and do not by themselves certify compliance [3].

CHAPTER 1

Why a Type 2 report needs continuous evidence

SOC 2 examinations use the AICPA Trust Services Criteria, the 2017 criteria with points of focus revised in 2022 [1]. A Type 1 report looks at whether controls are designed well at a point in time. A Type 2 report also tests whether they operated across an observation period, often six to twelve months. The auditor samples from that whole period. If a log, an alert or a review from month two is missing, the control has a gap, however good it looks today.

So the evidence system has three jobs. It must **collect** the right events from every system in scope. It must **keep** them for longer than the observation period. And it must **preserve** monthly object versions under retention controls, with separate restrictions on privileged storage administrators. Collection alone is insufficient. Retention and protected monthly records also need configuration and regular checks.

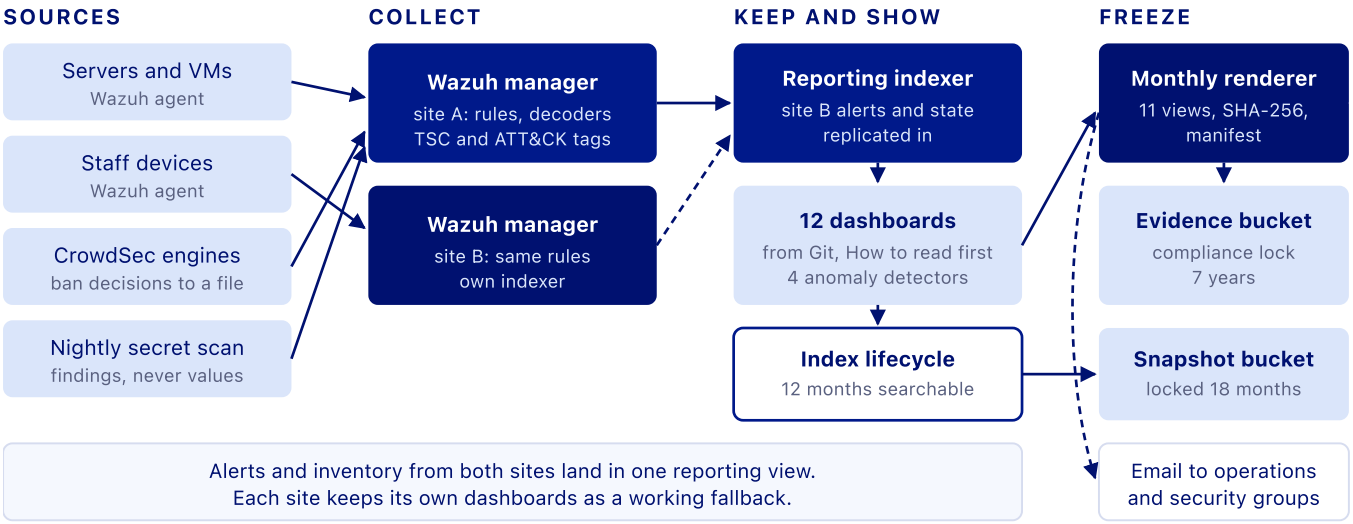


Figure 1. Collect from every source, keep it longer than the audit window, and retain monthly object versions under controlled storage access.

Design choices that matter

- **Servers and people.** Staff laptops and desktops run the same agent as servers. Many access and change controls are about people.
- **Two sites, one view.** Each site has its own manager and indexer, so a site outage does not stop collection. Alerts and inventory state replicate into one reporting indexer.
- **Dashboards as code.** Every dashboard, detector and report setting lives in Git and is deployed by a playbook. A dashboard edited by hand is exported and committed, or it does not exist.
- **Same rules at every site.** Both managers load the same custom rules and decoders from Git, so an alert means the same thing wherever it fired.
- **Least privilege for evidence.** The report renderer signs in as a read-only user. It can read every dashboard and change none of them.
- **Pinned versions.** The Wazuh version, the renderer image and every helper container are pinned. An upgrade is a reviewed change, and the report looks the same from one month to the next.

CHAPTER 2

Mapping Wazuh to the Trust Services Criteria

Wazuh tags many of its rules with the Trust Services Criteria they relate to, and documents how log collection, file integrity monitoring, configuration assessment, threat detection and vulnerability assessment support them [3]. Rule tags alone are not a mapping an auditor can follow. We built one dashboard per control area and wrote the criteria into each dashboard's description. The monthly report reads those descriptions and prints them on every page.

Criterion	What it asks, in short	Evidence from Wazuh
CC3.2	Identify and analyse risks to objectives	Open vulnerabilities by severity, host and package [4]
CC6.1	Logical access security over information assets	Logins, failed logins, secrets found in source code
CC6.2, CC6.3	Register, authorise, change and remove access	New users and groups, account changes, privileged commands
CC6.8	Prevent or detect unauthorised or malicious software	Rootcheck and malware detections, possible rootkit alerts
CC7.1	Detect configuration changes and new vulnerabilities	CIS benchmark results [6], file integrity changes, vulnerability state
CC7.2	Monitor components for anomalies	All alerts by severity, agent coverage, anomaly detectors
CC7.3, CC7.4	Evaluate security events and respond to incidents	Level 10 and higher alerts with MITRE ATT&CK tactics [5]
CC8.1	Authorise, test and implement changes	Package installs and removals, configuration and policy changes

Table 1. The criteria our dashboards support. Wording is paraphrased. Read the AICPA text for the full criteria [1].

Some Wazuh rules also carry availability and processing integrity criteria, such as A1.2, PI1.4 and PI1.5. They appear on the overview dashboard. We have not built evidence around them yet, because our first report covers the security criteria.

Rules for an evidence dashboard

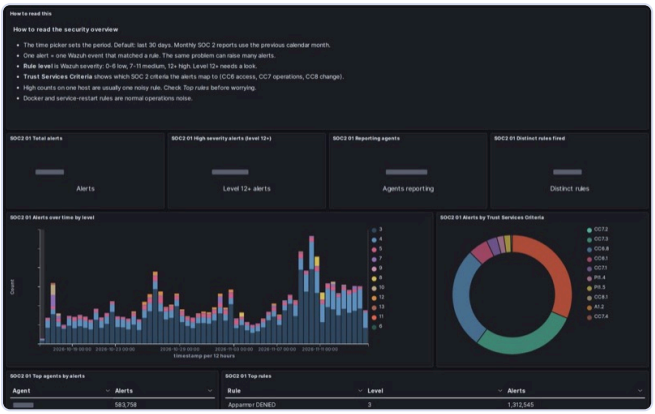
- **Open with "How to read this".** A full-width panel of five to eight short lines: what period the numbers cover, what one unit counts, what normal noise looks like and what to act on. Leadership asked for it on every dashboard after the first one turned alarm into understanding.
- **Say whether it is a period or a snapshot.** Alert dashboards follow the time picker. Vulnerability dashboards show the state on the day they are rendered. The panel says which.
- **Name the noise.** Automation accounts, health checks, scanners on the internet and your own egress addresses all create alerts. List them so a reviewer does not chase them.
- **Put criteria in the description.** One line such as "CC6.1, CC6.2, CC6.3: logins, failed logins, privilege use and account changes". The report cover builds its index from it.
- **End in a decision.** Every table answers "what do I do next": rotate, patch, triage or explain.
- **Keep one owner.** A playbook owns each dashboard and detector. An edit made only in the browser is overwritten on the next run unless it is exported to Git first.

> **Coverage is a control too.** A dashboard that shows no failed logins is only evidence if every system was reporting. That is why agent coverage has its own dashboard, and why disconnected servers need follow-up the same day.

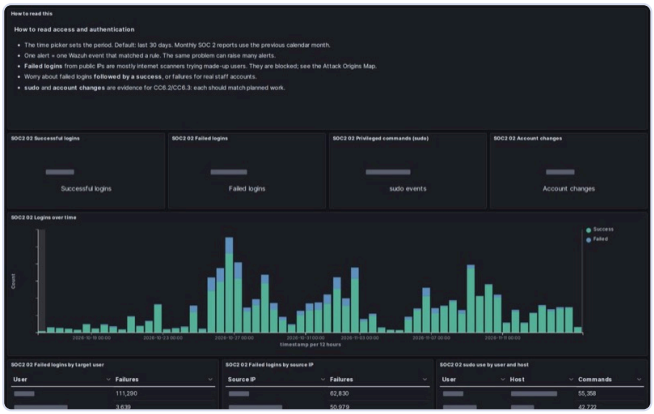
CHAPTER 3

The twelve dashboards, as they run today

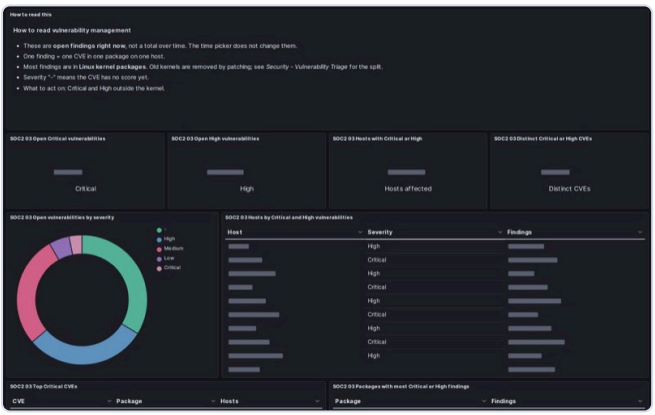
These are November 2026 captures from our deployment. Event views use a 30-day window. Vulnerability views show the state on capture day. Host names, user names, addresses, packages, file paths, repository names and headline totals are masked. The layout, panels and "How to read this" text are real.



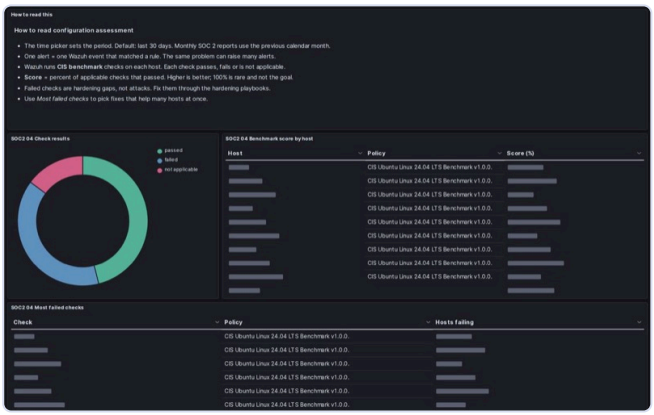
1. Security overview CC7.2, CC7.3. All alerts by severity, criteria, host and rule.



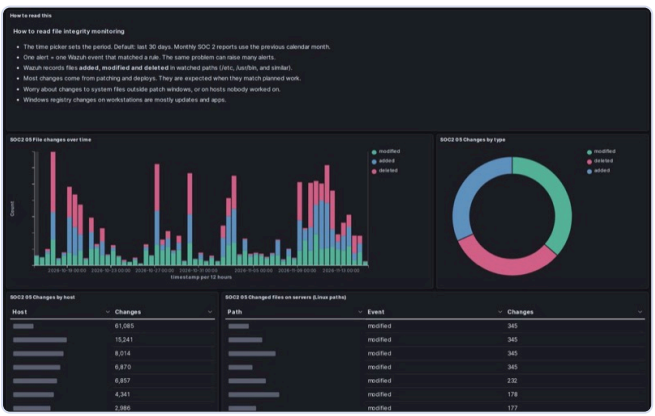
2. Access and authentication CC6.1 to CC6.3. Logins, failures, sudo use and account changes.



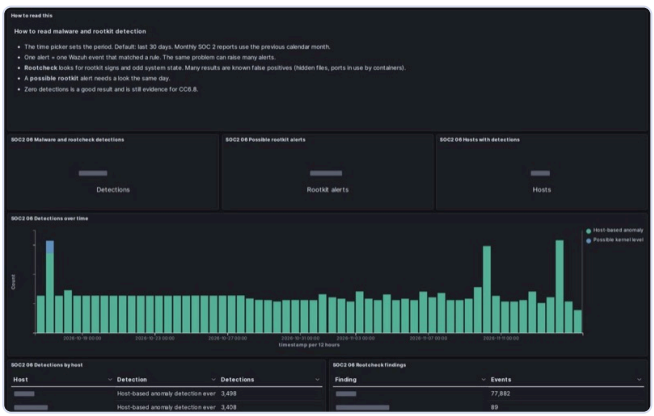
3. Vulnerability management CC7.1, CC3.2. Open Critical and High findings today.



4. Configuration assessment CC7.1, CC8.1. CIS benchmark results and score per host.



5. File integrity monitoring CC8.1, CC7.1. Files added, modified and deleted in watched paths.

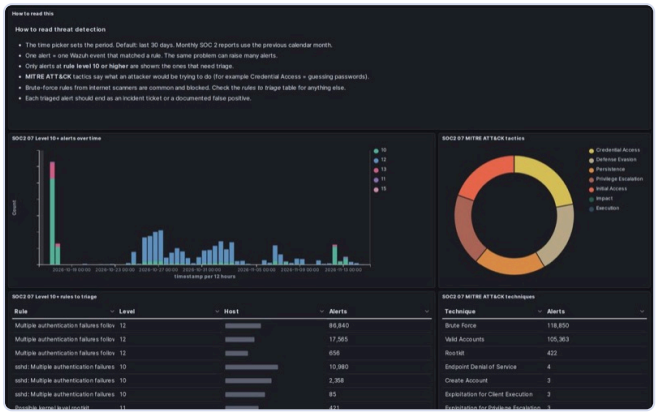


6. Malware and rootkit detection CC6.8. Rootcheck and malware results. Zero is still evidence.

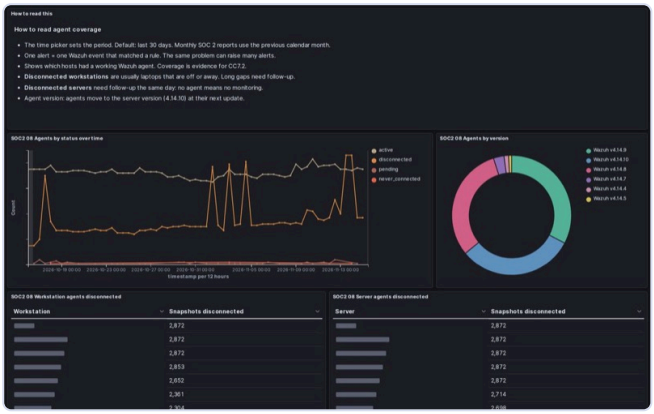
CHAPTER 3, CONTINUED

Triage, coverage, change and three security views

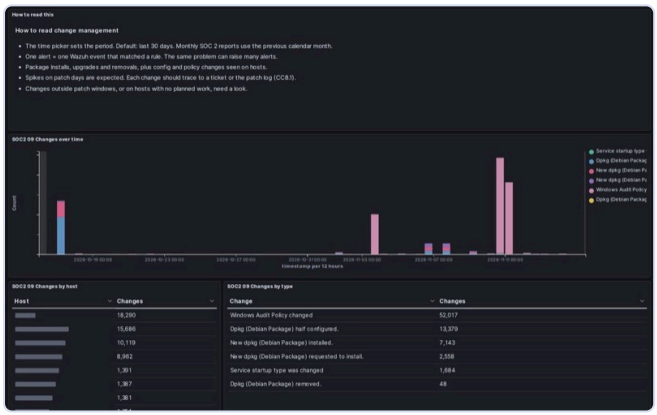
Dashboards 7 to 9 complete the SOC 2 set. Three security views add context. The monthly pack includes dashboards 1 to 11. Secrets Exposure is available for daily review but is not yet scheduled in the pack.



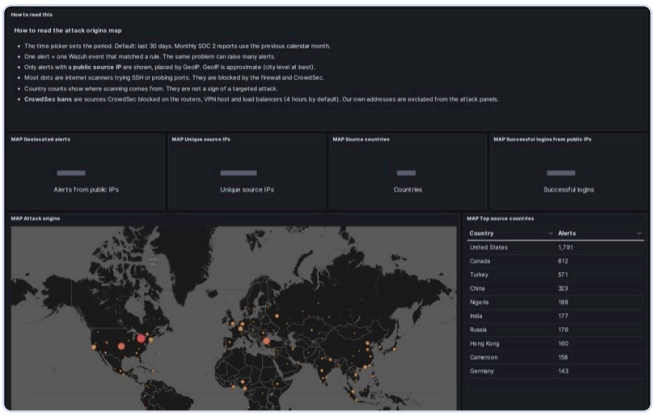
7. Threat detection and triage CC7.3, CC7.4. Level 10 and higher alerts with ATT&CK tactics.



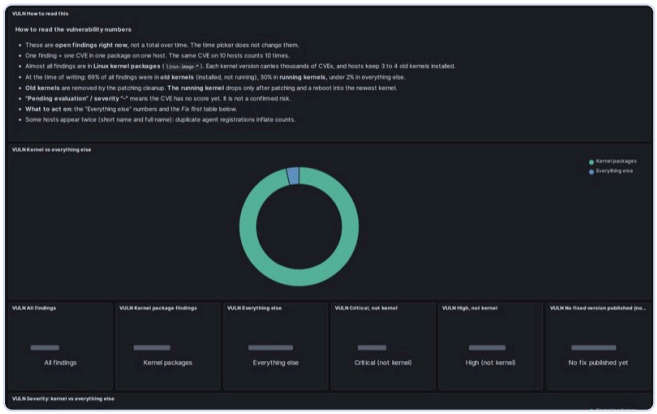
8. Agent coverage CC7.2. Agents by status and version, disconnected servers and devices.



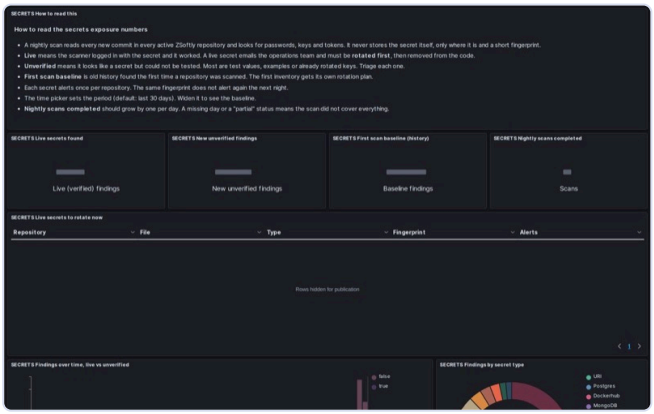
9. Change management CC8.1. Package installs and removals, policy changes.



10. Attack origins map Supporting. Public-source alerts by country, CrowdSec bans, logins to review.



11. Vulnerability triage Supporting. Splits kernel noise from what needs action.



12. Secrets exposure CC6.1, CC7.1. Secrets found in source code, live ones first.

CHAPTER 4

The monthly evidence pack and retention

Dashboards answer questions today. Auditors ask about last March. Historical event reports and retained logs support that review. Vulnerability reports instead preserve the state on the day of generation.

How the monthly pack is built

1. A timer runs monthly at 06:30 local time on day one. Nine event views cover the previous calendar month. Two vulnerability views show run-day state.
2. A pinned headless browser opens the 11 scheduled dashboards as a read-only user. Its container has a read-only root file system and no Linux capabilities. Secrets Exposure is excluded.
3. It waits until every panel reports that it has rendered, then prints one PDF per dashboard with a header that carries the criteria, the period, the time of generation and the source. A cover page lists every dashboard and its criteria.
4. The job writes a SHA-256 hash for every file and a manifest that records the container images, script hashes, object keys, version IDs and lock dates.
5. It uploads the PDFs and checks that each stored version carries a compliance-mode lock. The manifest goes last.
6. It emails the pack to the operations and security groups. A failed run sends a failure email and sets a monitoring metric instead.

Why compliance-mode object lock

Compliance mode protects a retained object version from deletion or a shorter retention period through the S3 API [9]. Our Ceph gateway implements these calls [10], with seven-year report retention. Privileged Ceph administration outside the S3 API remains a separate access control. Delete markers might hide an object, so the manifest records version IDs for retrieval. Hashes help detect changes.

Log retention: 12 months searchable, 18 months cold

For log retention we adopted the split in US federal memo M-21-31: 12 months in active storage and 18 months in cold data storage [2]. It is longer than most SOC 2 windows and gives room to investigate an incident found late in a period.

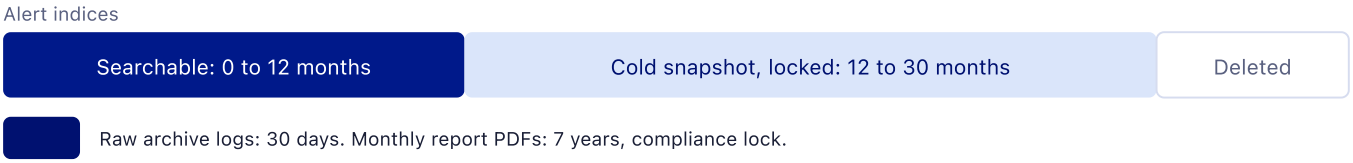


Figure 2. An index lifecycle policy snapshots each daily index to a locked bucket at 12 months, then deletes it.

The OpenSearch index lifecycle feature runs these steps by index age [7]. The snapshot bucket holds each snapshot under an 18-month compliance lock. Every quarter our security engineer restores one snapshot into a scratch index and compares document counts with the source. The first test matched every document. A backup you have never restored is a hope, not a control.

CHAPTER 5

Signals beyond the agent

These integrations add context to host monitoring. Secrets Exposure is available for daily review but is not yet in the 11-dashboard monthly pack. Dependency findings stay in GitHub.

CrowdSec bans as threat intelligence

CrowdSec runs on routers, the VPN host and load balancers. Its file notification plugin provides a Wazuh-compatible alert format [11]. Custom rules record bans with their scenario and source country. The attack origins map excludes our own egress addresses to avoid misleading GeoIP results.

Anomaly detection

OpenSearch scores unusual data points with Random Cut Forest [8]. Four detectors watch 10-minute buckets: total alerts, failed logins and distinct sources, alerts at level 10 or higher, and alerts per host. We allow a week of learning before enabling notifications.

Vulnerability triage

Wazuh matches software inventory against vulnerability data [4]. A finding counts one CVE, package and host. In our review, 69 percent were in old, inactive kernels and 30 percent in running kernels. Under 2 percent were elsewhere. Triage shows this split. Monthly patching removes old kernels on most hosts.

Secret and dependency scanning

Layer	Tool	What it does for us
Prevent, private repositories	Gitleaks, MIT licence [13]	Push and pull-request checks in participating active private repositories use a shared workflow with a pinned, checksum-verified scanner
Prevent, public repositories	GitHub push protection [14]	On enabled public repositories, blocks pushes containing supported secret patterns
Detect, every night	TruffleHog, AGPL-3.0 [12]	Scans new commits in repositories included in the nightly scan and attempts to verify detected credentials
Evidence	Wazuh custom rules	Records location and fingerprint, never the value. Verified secrets trigger high-severity email alerts. Findings appear in Secrets Exposure

Table 2. Secret scanning in layers. It supports CC6.1 for protected credentials, CC7.1 for vulnerability detection and CC7.2 for monitoring and alerting.

Exclusions need an owner and reviewed pull request. Scan history survives repository renames. An isolated runner group prevents public repositories from using the scanner. Review the completed-scans panel daily for missed runs.

Dependency vulnerabilities. Dependabot flags vulnerable dependencies in enabled repositories [15]. Version updates open pull requests for eligible updates on configured schedules [16]. These findings remain in GitHub, outside Wazuh and the monthly pack.

CHAPTER 6

Lessons from production and a readiness checklist

What happened	Cause	What we changed
Alert data was days from being deleted halfway through the window	The lifecycle policy kept indices for 180 days, a default nobody had compared with the observation period	12 months searchable plus 18 months locked cold storage, set in code and checked first in any SOC 2 work
Vulnerability dashboards covered a minority of hosts	Each site wrote inventory state to its own indexer. The reporting view only saw one site	Cross-site replication of state indices, and a coverage check on the denominator before trusting any total
The owner saw large numbers and assumed a crisis	Raw totals with no unit or baseline. Kernel packages dominated every count	A "How to read this" panel on every dashboard, a triage view, and kernel clean-up in patching
The map printed with no points	The page was saved before the map layer finished drawing its data	Wait for data on screen, not only a status flag, and look at the first PDF of every new dashboard
A nightly scan did not run and nothing complained	A schedule fails silently. Success was the only thing measured	A completed-scans panel that must grow by one a day. A missing-run alert is next
A storage key appeared in a setup log	A validation step printed the variables it checked	Keys rotated before first use, secrets hidden from task output, the leaked log deleted
Some hosts were counted twice	A host re-registered under a new name kept its old record	A duplicate check in the coverage review before any count is trusted

Table 3. Anonymised lessons from building this. Each fix is now code, a dashboard panel or a review step.

Check your readiness

- [ ] Every server and staff device in scope runs an agent.
- [ ] Disconnected servers are followed up the same day.
- [ ] Each dashboard names its criteria and opens with "How to read this".
- [ ] Dashboards and detectors live in Git and deploy from a pipeline.
- [ ] Log retention is longer than your observation period.
- [ ] Cold copies are restored and counted every quarter.
- [ ] A monthly pack is rendered without a person in the loop.
- [ ] Each pack has hashes, retained version IDs and restricted storage administration.
- [ ] A failed run alerts a person as well as writing a log.
- [ ] Known noise is listed: automation accounts, scanners, own addresses.
- [ ] Secrets are blocked before commit and scanned every night.
- [ ] Someone reviews each pack and files gaps as findings.

Address collection and retention gaps first. Assign an owner and deadline to each unchecked item, then verify completion. Measure agent coverage and retained data before trusting dashboard totals.

CHAPTER 7

Where Wazuh stops

Wazuh gathers technical evidence. A SOC 2 examination also asks for things no security agent can see: approved policies, staff acknowledgements, onboarding checks, security training records, vendor reviews, a risk register, and answers to the auditor's evidence requests. That is the governance, risk and compliance layer. Commercial compliance platforms sell software for it.

Evidence Wazuh can produce

- Logins, privilege use and account changes
- Vulnerabilities, configuration and file changes
- Malware, threat triage and agent coverage

Evidence that needs people and process

- Policies, approvals and staff acknowledgements
- Onboarding checks and training completion
- Vendor reviews, risk decisions and audit requests

How we run that layer today

- **Nine tracked workstreams.** Access control, change management, incident response, risk, vendors, continuity, policies, technical evidence and audit readiness. Each has written scope and acceptance criteria.
- **A control matrix in progress.** We are building a matrix in Git with each control's criteria, owner, evidence source, collection frequency and retention. Missing entries will identify work still to document.
- **Evidence that already exists.** Runbooks, postmortems, monthly patching records and the user lifecycle procedure live in our documentation repository with full history.
- **Still being written.** The formal policy set, staff acknowledgements, training records, the vendor inventory and the risk register. They will sit in the same repository with an owner, a version and an approval date.
- **An open decision.** Whether to add a compliance platform or keep collecting in-house is not decided. We will record the reason either way.

Build the evidence first. A compliance platform can show you what is missing. We remain responsible for log retention, server patching and checking whether every host is reporting. Teams that put the technical evidence in place first get more out of any platform they choose later.

How ZSoftly can help

ZSoftly Technologies Inc. is a cloud engineering company. We design, build and run cloud platforms for clients in Canada, the United States and the United Kingdom. The engineers who build your evidence pipeline are the ones who hand it over.

Assess	Map	Build	Operate	Hand over
Scope, current logging, retention and coverage gaps.	Agree criteria, dashboards and noise lists in writing.	Agents, dashboards as code, reports and locked storage.	Run the monthly review with your team.	Code, runbooks and keys move to your team.

Get SOC 2 audit-ready before you buy a compliance platform. Talk to our engineers. Book a discovery call at zsoftly.ca/contact or call +1 (647) 771-1065. Your first hour with our engineers is free.

REFERENCES

Sources

All sources were checked on 15 November 2026.

1. AICPA. 2017 Trust Services Criteria (With Revised Points of Focus, 2022). https://www.aicpa-cima.com/resources/download/2017-trust-services-criteria-with-revised-points-of-focus-2022
2. US Office of Management and Budget. M-21-31, 27 August 2021, Appendix C, logging requirements and retention. https://www.whitehouse.gov/wp-content/uploads/2021/08/M-21-31-Improving-the-Federal-Governments-Investigative-and-Remediation-Capabilities-Related-to-Cybersecurity-Incidents.pdf
3. Wazuh. Regulatory compliance: Trust Services Criteria. https://documentation.wazuh.com/current/compliance/tsc/index.html
4. Wazuh. Vulnerability detection. https://documentation.wazuh.com/current/user-manual/capabilities/vulnerability-detection/index.html
5. MITRE. ATT&CK knowledge base. https://attack.mitre.org/
6. Center for Internet Security. CIS Benchmarks. https://www.cisecurity.org/cis-benchmarks
7. OpenSearch. Index State Management. https://docs.opensearch.org/latest/im-plugin/ism/index/
8. OpenSearch. Anomaly detection. https://docs.opensearch.org/latest/observing-your-data/ad/index/
9. AWS. Locking objects with S3 Object Lock, retention modes. https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html
10. Ceph. Object Gateway S3 API, bucket operations and object lock. https://docs.ceph.com/en/latest/radosgw/s3/bucketops/
11. CrowdSec. File notification plugin. https://docs.crowdsec.net/docs/local_api/notification_plugins/file/
12. Truffle Security. TruffleHog. https://github.com/trufflesecurity/trufflehog
13. Gitleaks. https://github.com/gitleaks/gitleaks
14. GitHub. About push protection. https://docs.github.com/en/code-security/secret-scanning/introduction/about-push-protection
15. GitHub. About Dependabot alerts. https://docs.github.com/en/code-security/dependabot/dependabot-alerts/about-dependabot-alerts
16. GitHub. About Dependabot version updates. https://docs.github.com/en/code-security/dependabot/dependabot-version-updates/about-dependabot-version-updates

This paper is for general information. It is not audit or legal advice and creates no warranty or commitment from ZSoftly Technologies Inc. SOC 2 is a registered trademark of the AICPA. Wazuh is a trademark of Wazuh, Inc. ATT&CK is a registered trademark of The MITRE Corporation. CIS Benchmarks is a registered trademark of the Center for Internet Security. Other names may be trademarks of their owners. © 2026 ZSoftly Technologies Inc. Version 1.0, November 2026.