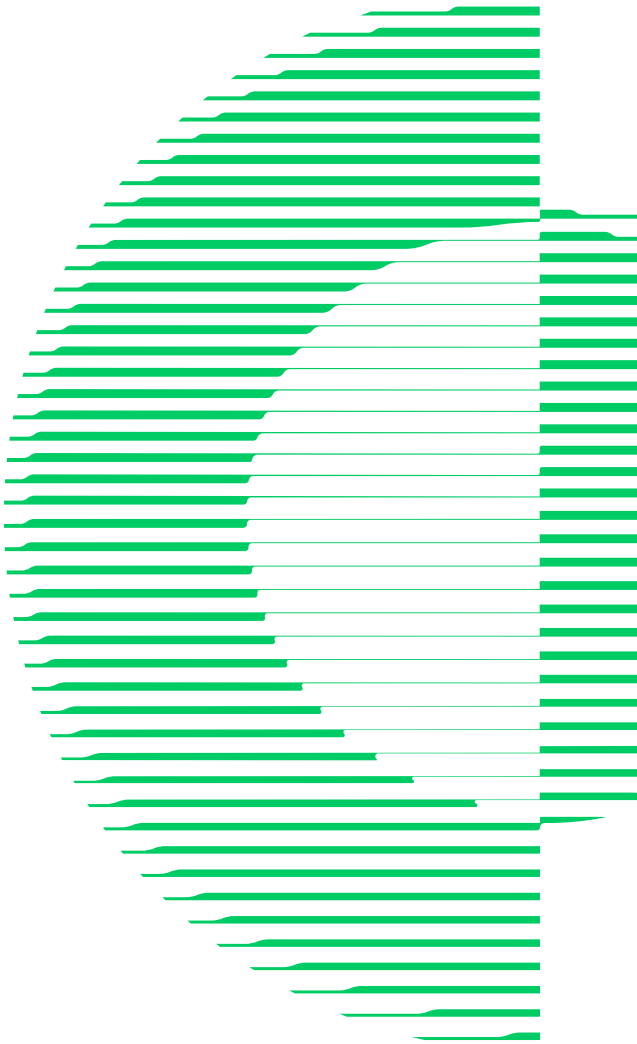
paloalto NETWORKS | CORTEX® CLOUD

The Security Checklist for Cloud Defenders

Cloud environments are ephemeral by nature. Static controls lose relevance. Permissions drift. Tools evolve faster than governance frameworks can keep up. *The Security Checklist for Cloud Defenders* fuels security discussions and helps ground strategic reviews in current conditions.

Table of Contents



How to Use This Checklist

Each quarter, assign domain leads to revisit the sections relevant to their responsibilities. Use the *What to Evaluate* prompts to anchor structured reviews. Reference the *Indicators of Success* to validate outcomes. The *Action Items* serve as checks on execution, evaluating whether ownership remains clear, automation still works, and controls still align with business priorities.

Build confidence through evidence. Logs, config snapshots, compliance states, and system behavior should support every assertion of control. Replace verbal assurance with telemetry. Prioritize domains where posture has drifted, threats have changed, or confidence is low.

Score each area on readiness. Don't aim for perfection across the board. Focus remediation where it matters—on critical workloads, demonstrable gaps, or high-blast-radius risks. The checklist doesn't measure completeness. Its goal is to enable operational assurance that can hold up to scrutiny.

Identity and Access Foundations

Every cloud security program depends on clear identity boundaries and hardened access practices. Identity remains the most targeted and most misconfigured control plane in the cloud. A zero trust requires teams, without exception, to enforce who can access what, from where, and under what conditions.

What to Evaluate

- Do we apply MFA consistently across all cloud identities, including third-party integrations and service accounts?
- When was the last comprehensive review of IAM roles and policies across cloud providers?
- Are there stale tokens or unrotated keys, lingering access paths that could allow privilege escalation?
- Have we limited the use of wildcard permissions (e.g., access policies) across our environment?
- Can we trace the creation and activity of every privileged action back to a verified identity?

Control Maturity Grid — Identity and Access Hygiene

Capability	Basic (Ad Hoc)	Intermediate (Policy-Based)	Advanced (Automated and Enforced)
MFA Coverage	Admins only; no enforcement for services	MFA required for all human users	Enforced via IdP; includes federated and service accounts
Privileged Access Reviews	Manual, irregular audits	Quarterly reviews with manual attestations	Automated reviews; integrated with JIT access workflows
Key and Token Rotation	No expiration or alerts on static credentials	Manual tracking; keys rotated by owner discretion	Automated rotation and expiry enforcement
Wildcard Permissions	Common in policy definitions	Policies scoped per role; flagged	Blocked by policy as code
Access Logging and Attribution	Partial logging; limited user correlation	Full logging with manual mapping	Blocked by policy as code

Action Items

1. **Enforce MFA everywhere:** Require MFA for all human and service identities through IdP policy. Monitor for enforcement drift.
2. **Eliminate stale accounts:** Identify accounts inactive for 90+ days. Remove or reclassify with documented business justification.
3. **Audit admin access paths:** Map all privilege escalation routes. Log every use of elevated access and require human attestation.
4. **Restrict overbroad roles:** Disallow wildcard permissions and enforce least privilege at the group and role level using policy-as-code enforcement.
5. **Rotate keys automatically:** Set max lifespan for all static credentials. Alert or autorevoke credentials that exceed age thresholds.

Success Indicators

- ✓ **200% MFA enforcement across all identities**
  Includes human, federated, and service identities. Enforced by policy, not configuration drift. Verified through monthly attestation.
- ✓ **Less than 1% of identities classified as stale**
  No more than 1% of accounts show no activity in 90+ days without documented exception and review. Tracked through automation.
- ✓ **Zero wildcard privileges in production**
  All IAM permissions scoped to least-privilege. Environment-wide scans surface no use of *:* policies in live infrastructure.
- ✓ **Privileged access audits completed quarterly**
  Every quarter, teams complete access reviews that trace high-risk roles to current users. Outcomes are documented and drive permission pruning.
- ✓ **Privileged actions are mapped to identity with <1% ambiguity**
  Audit trails attribute high-sensitivity changes (e.g., policy updates, privilege escalations) with full fidelity. Anomalies are reviewed and remediated within SLA.
- ✓ **Static credentials older than 30 days are automatically rotated or revoked**
  Key rotation policies are actively enforced. Alerts fire on violation, and teams act without prompting.
- ✓ **Root account usage is <0.1% of all cloud activity**
  Root accounts are disabled where possible. Where allowed, usage is rare, logged, and governed by strict just-in-time access procedures.
- ✓ **Access governance metrics are available on demand**
  Dashboards display identity inventory, privilege distribution, access recency, and key rotation posture.

Data Discovery and Classification

Fragmented data environments create blind spots. Inventories don't reflect risk without a deep understanding of data type, classification, and exposure.

What to Evaluate

- Have we identified every location where sensitive data is stored, including unmanaged and ephemeral services?
- Can we distinguish between regulated data (e.g., PCI, HIPAA, GDPR) and business-critical data?
- Are we classifying data dynamically as it moves or changes structure?
- Do we apply consistent tagging and labeling across accounts, buckets, and storage services?
- Have we validated that no unencrypted sensitive data exists in test or backup environments?

Encryption and Key Management

Encryption without control creates a false sense of security. Teams must manage not just encryption at rest or in transit, but also key ownership, usage, and auditability.

- Are we using customer-managed keys (CMKs) for all sensitive workloads, rather than relying on provider-managed defaults?
- How often do we rotate encryption keys? Can we verify that the rotation was successful and complete?
- Have we segmented key access by role and function? Can any identity escalate into key management without review?
- Are audit logs enabled for all encryption key operations, including failed attempts?
- Do we test decryption workflows as part of incident response planning?

Control Maturity Grid — Data Discovery and Classification

Capability	Baseline Practices	Improving Practices	Mature Practices
Discovery Coverage	Discovery efforts focus on known storage systems or specific cloud services.	Automated discovery tools scan major environments, but gaps remain in ephemeral assets.	Discovery runs continuously across cloud services, regions, and ephemeral resources.
Data Sensitivity Identification	Sensitivity is inferred manually or through ad hoc tagging.	Data is classified using rule-based scanners for regulated data types.	Classification combines pattern recognition, context, and ML-based inference at scale.
Labeling and Tag Consistency	Labels are inconsistently applied across buckets, databases, and services.	Standardized labeling exists for regulated data, but adoption varies.	Labeling is enforced by policy as code and validated across the full cloud environment.
Unmanaged Data Detection	Shadow datastores and test environments are rarely assessed.	Drift is detected through periodic reassessment of labeled data.	Classification status is monitored in real time, with alerts on policy violations or mislabeling.
Classification Drift Monitoring	Classification accuracy isn't tracked or reevaluated.	Periodic scans include staging and backup systems.	Continuous discovery detects unencrypted or unmanaged data, including in dev and transient systems.
Access-Aware Classification	Classification operates independently of access context.	Role- and group-based access is partially aligned with sensitivity labels.	Access and sensitivity are evaluated jointly to detect overexposure or misuse of sensitive data.

Action Items

1. **Scan for unclassified data:** Use DSPM tools or CSP-native functions to identify sensitive data across object stores, databases, and ephemeral storage.
2. **Classify and label data consistently:** Define and enforce a classification schema through policy as code. Require labels on deployment, not post hoc.
3. **Encrypt data with CMKs:** Transition critical data to customer-managed keys. Validate CMK enforcement using access logs and policy enforcement metrics.
4. **Rotate keys on schedule:** Automate key lifecycle management. Confirm downstream compatibility and detect anomalies in rotation execution.
5. **Limit key access by role:** Harden IAM policies. Require just-in-time access, justification logging, and real-time alerting for key-related operations.

Success Indicators

- **100% of sensitive data assets are classified and tagged**
  Automated discovery tools detect and label all regulated and business-critical data. Classification schemas apply across object storage, databases, and ephemeral volumes.
- **No unencrypted sensitive data in any environment**
  Scans surface zero instances of sensitive data (e.g., PII, PHI, payment data) stored unencrypted at rest or in transit. All test, backup, and analytic environments comply.
- **≥ 95% of encryption uses customer-managed keys**
  Workloads containing sensitive data rely on CMKs or HSM-backed CMKs. Provider-managed default keys are fully deprecated for regulated data.
- **Key rotation completed on time for 100% of active CMKs**
  Key rotation schedules are enforced by automation. Each rotation event triggers verification to confirm continuity across dependent services.
- **All key access paths are role-restricted with zero exceptions**
  IAM boundaries prevent unauthorized access to encryption keys. No user or service account has administrative control outside of documented workflows.
- **Audit logs cover 100% of key lifecycle operations**
  All key usage, creation, deletion, and access attempts, whether successful or failed, are logged and retained. Logs integrate with SIEMs and drive detection rules.
- **Decryption workflows are tested at least twice per year**
  Incident response exercises validate that teams can retrieve and decrypt protected data reliably. Tests cover multiple data classes and include real key recovery.
- **Data protection dashboards reflect live inventory and coverage gaps**
  Security teams can display encryption posture, classification completeness, and key ownership by business unit, sensitivity, and service type without manual queries.

Runtime Security and Workload Protection

Once deployed, workloads stop behaving like static assets. They spawn ephemeral processes, cross trust boundaries, and generate behaviors that scanners can't predict. Static controls, even when well-informed upstream, can't verify whether workloads execute as intended. Runtime security must confirm that deployed code honors the posture it passed through build and deploy, and intervene when it doesn't.

What to Evaluate

- Do we block vulnerable or misconfigured images before deployment?
- Can we detect behavioral drift or anomalous activity from containers and serverless functions in production?
- Are kernel-level protections such as eBPF, AppArmor, or seccomp in place for all production workloads?
- Have we isolated production workloads by trust level to prevent lateral movement?
- Are we capturing and analyzing runtime telemetry (e.g., syscall traces, DNS activity, process trees)?

Automation Readiness Grid — Runtime Controls by Workload Type

Workload Type	Predeployment Enforcement	Runtime Monitoring	Behavioral Detection	Automated Containment
Container Images	SCA/IaC scan on build; advisory only	Logging enabled; no baselining	Drift detection via manual tuning	Manual blocking via alert response
Containers	Blocking scan policies; policy-as-code enforced	Continuous monitoring with baselining	eBPF-driven anomaly detection	Auto quarantine on verified triggers
Serverless	Linter-only or partial scanning	Limited or no telemetry; inspect network activity	No runtime baselining; network anomaly detection	Autoblock risky network activity (or manual disable)
VMs (Prod)	OS patching policies in CI	EDR with host telemetry	Process and file integrity monitoring	Auto ticketing and lockdown via SOAR

Action Items

1. **Scan every workload at build:** Configure CI pipelines to enforce blocking policies on images, IaC templates, and function code. Fail builds that introduce high-severity vulnerabilities or misconfigurations.
2. **Instrument runtime with eBPF or equivalent:** Enable fine-grained syscall monitoring across containers and hosts to detect unexpected behavior without injecting agents.
3. **Baseline and alert on drift:** Build behavioral baselines for workload execution patterns. Trigger investigations when processes, filesystem activity, or outbound traffic diverge from the norm.
4. **Segment workloads by trust:** Enforce environment segmentation to keep dev, staging, and production workloads isolated. Apply per-segment policies for network, access, and logging.
5. **Test autocontainment logic:** Simulate known attack patterns and verify that detection-to-response handoffs (e.g., container pause, function disable, VM lockdown) fire correctly.

Success Indicators

- **100% of production workloads scanned before deployment**
  All container images, serverless functions, and VM templates undergo automated security scanning in CI/CD pipelines. Blocking policies prevent promotion of high-risk artifacts.
- **Runtime monitoring deployed across all active workloads**
  Every running container, function, and virtual machine reports live telemetry to a central platform. No production workload operates outside of coverage.
- **Baseline drift alerts triggered and reviewed within SLA**
  Behavioral anomalies generate alerts. Each alert links to a documented investigation within the defined response time-frame.
- **Syscall restrictions enforced across ≥ 90% of containers**
  Production containers run with enforced seccomp, AppArmor, or equivalent profiles. Drift or bypass attempts are logged and blocked in real time.
- **No untracked container images in production**
  All container images in use trace back to verified builds. Shadow images and outdated versions are automatically flagged and quarantined.
- **VMs enrolled in EDR with full visibility**
  100% of persistent virtual machines participate in endpoint detection and response programs. Telemetry includes process-level behavior, integrity checks, and remote session tracking.
- **No reused base images across environments with different trust levels**
  Base images reflect environment-specific hardening requirements. Image reuse across dev, staging, and prod triggers enforcement rules and approval workflows.
- **Runtime metrics integrated into cloud security dashboards**
  Dashboards provide real-time visibility into scan coverage, runtime anomalies, baseline drift, and workload segmentation. Security and platform teams use this data for daily triage and incident prevention.

Network and Perimeter Defense

Cloud networks fracture traditional security perimeters, and identities replace IPs. The illusion of isolation often hides the reality of lateral exposure. Security teams must control what can reach the cloud and what can move inside it. Visibility and segmentation form the new firewall. Policy replaces patchwork.

What to Evaluate

Segmentation and Exposure Control

The cloud grants immediate connectivity across accounts, regions, and services. Without strict segmentation, workloads can reach each other with no meaningful boundaries. If defense relies on the assumption that all parts of the cloud are equally secure, then all parts are equally insecure. Every security boundary should be treated like its own perimeter. Effective segmentation contains the blast radius and forces lateral movement into monitored and controlled choke points.

- Do we default to deny on all ingress rules? Are exceptions reviewed and time-bound?
- Are our security groups scoped to the narrowest possible CIDR blocks and ports?
- Can we identify every public IP or exposed service across all cloud accounts?
- Do we alert on unexpected egress behavior from workloads and classify external destinations to streamline anomaly detection?
- Have we isolated management plane access from general workload traffic?

Zero Trust Network Architecture

In a zero trust model, no request earns implicit trust. Every connection must prove its identity, and every action must be authorized in context. In cloud networks, that requires workload identity, encrypted service-to-service communication, and continuous validation across internal network flows.

- Are all internal service communications encrypted with mutual TLS or a service mesh?
- Do we authenticate internal API calls with workload identities, not just network presence?
- Have we decoupled trust from network location across availability zones and regions?
- Can we audit who, or what, initiated each internal network connection?
- Have we implemented policy engines to evaluate network access contextually in real time?
- Are we monitoring network connection payloads for potentially malicious content?

Strategic Grid – Network and Perimeter Defense

Capability Area	Segmentation Coverage	Exposure Control	Identity-Aware Access	Audit Readiness
External Web Applications	Isolated in DMZ	Deny by default; audited quarterly	AuthN + WAF session inspection	Compliant; last reviewed [current month]
Internal API Gateways	Scoped to service tier	Ingress rules reviewed; alert on deviation	Workload ID via service mesh	Compliant; alert logs enabled
East/West connectivity	Security controls between segments	Deny by default; traffic payload inspected for threats and malware	Orchestrated policies end to end	Compliant; full logging enabled
Management Plane Interfaces	Restricted by jump host	IP-restricted; mTLS required	Role-scoped + device posture checked	Secure, policy-drive, transparent, compliant
Data Layer Resources	VPC-segmented	Private endpoint only	IAM policy-bound access only	Unknown; logging not enforced

Action Items

1. **Deny by default:** Configure all security groups, firewalls, and access control lists to reject all inbound traffic unless explicitly approved. Review rules quarterly.
2. **Segment by trust level:** Group workloads by function and sensitivity. Use subnet-level segmentation, VPC peering policies, and security zones to restrict internal paths.
3. **Audit external exposure:** Inventory every asset with a public IP or inbound path from the internet. Decommission or rearchitect those not essential to external workflows.
4. **Inspect East-West traffic:** Detect threats and malware with network security controls for traffic between segmented security boundaries.
5. **Encrypt internal traffic:** Deploy mTLS through service meshes or mutual authentication proxies to protect east-west traffic. Validate certificate rotation and key integrity.
6. **Enforce identity-aware access:** Require strong identity verification for every network request. Evaluate conditions such as role, device posture, and request timing before allowing access.

Success Indicators

- ✓ **100% of ingress rules default to deny**
  Every security group and firewall policy starts from an explicit deny baseline. Exceptions are time-bound, reviewed quarterly, and scoped to verified need.
- ✓ **All internet-facing assets are known and documented**
  Every publicly routable IP, open port, or exposed endpoint is captured in a live asset inventory. Unknown exposures trigger automated alerts and require remediation within SLA.
- ✓ **Internal network segmented by trust and function**
  Workloads are isolated by role, sensitivity, and environment. Lateral movement paths are constrained by enforced segmentation policies.
- ✓ **≥ 95% of internal service traffic encrypted in transit**
  All east-west communication between workloads—across zones, regions, and services—uses mTLS or equivalent encryption. Noncompliant traffic is logged, blocked, or rerouted.
- ✓ **Zero reliance on network location for access decisions**
  No access control depends solely on subnet, IP, or region. Policy engines assess contextual attributes (e.g., identity claims, device state) before granting access.
- ✓ **Internal network connections audited with full attribution**
  Audit logs attribute all internal connection attempts to verified workloads or identities. Unattributable flows are flagged for investigation and blocked by default.
- ✓ **Egress anomalies detected and triaged within SLA**
  Unexpected outbound traffic, especially to unknown destinations or ports, triggers immediate alerting and response. Logs support root cause analysis and policy refinement.
- ✓ **Choke points detect and block malicious traffic in real time**
  Traffic flowing between zones, tiers, or trust boundaries passes through monitored control points with enforced inspection, logging, blocking, and policy validation.

Threat Detection and Response

Cloud breaches begin with unnoticed activity that escapes or is lost in the noise of too much telemetry. Security teams must see early-stage signals across diverse systems, enrich them with context, and execute a defined, cloud-aware response. Detection without context leads to noise. Response without readiness leads to failure.

What to Evaluate

Telemetry and Logging

Cloud-native telemetry spans logs, metrics, traces, and posture data, which are often fragmented across services and formats. Without normalization and centralization, signals remain disjointed. Effective detection depends on ingesting high-fidelity telemetry from both first-party and third-party sources and transforming it into queryable, actionable insight.

- Are we forwarding audit logs (e.g., CloudTrail, GCP Admin Activity, Azure Activity Logs) to a central SIEM or data lake?
- Have we enabled detailed logging on all identity systems, storage services, and orchestration layers?
- Can we correlate logs across cloud services, SaaS applications, and identity providers?
- Are we normalizing logs to a common schema for search, detection, and forensics?
- Do we store logs long enough to support extended dwell-time investigations?

Detection Engineering

Threat actors rarely trigger known signatures. Cloud environments demand behavioral detection informed by workload context and enriched with posture data. Security teams must track identity misuse, anomalous privilege escalation, and interservice abuse. What's more, they must do so without drowning in false positives or brittle rule sets.

- Do we have detections tailored to our cloud architecture (versus general attack techniques)?
- Have we deployed anomaly-based detections that correlate posture with behavior?
- Can our detections identify stealthy attacks like token replay, overpermissioned API calls, or invisible data access?
- How quickly do we update detection logic based on new TTPs or incident learnings?
- Are alert thresholds tuned to reduce noise and focus response on high-fidelity signals?

Incident Readiness

Response playbooks must account for cloud-specific constraints—ephemeral infrastructure, identity-linked access, region-bound services, and distributed data. Waiting to translate traditional IR workflows into cloud response will cost time. Simulation and automation drive fluency.

- Do we maintain cloud-specific incident playbooks that account for service nuances and regional regulations?
- Have we prebuilt automated response actions (e.g., disable IAM role, quarantine VM, revoke token)?
- Do we simulate real-world attack scenarios that reflect current adversary behaviors?
- How quickly can we assemble complete context for any alert?
- Are we tracking metrics from past incidents to improve detection, triage, and containment time?

Control Maturity Grid – Threat Detection and Response

Capability	Manual or Ad Hoc	Automated and Policy-Enforced	Continuously Measured and Tuned
Audit log centralization and retention	Logs collected inconsistently across accounts; retention varies	Logs ingested into central SIEM with policy-backed retention	Retention, ingestion gaps, and log fidelity monitored via dashboards
Telemetry normalization and correlation	Each log source analyzed in isolation; limited cross-reference	Logs normalized into common schema with automated correlation	Correlation coverage scored across attack stages and asset types
Context-aware detection engineering	Basic detections rely on known indicators or static rules	Rules incorporate behavior, configuration, and asset context	Detection quality measured by coverage, fidelity, and noise rate

Action Items

1. **Enable and forward all audit logs:** Configure cloud-native logs (e.g., CloudTrail, Admin Activity Logs, Data Access Logs) and route them to a central SIEM or data lake with long-term retention.
2. **Normalize logs for unified analysis:** Adopt a common schema such as ECS or OCSF. Correlate across sources using timestamp alignment, identity resolution, and asset tagging.
3. **Deploy context-aware detections:** Develop rules that combine configuration, behavior, and identity context. Detect privilege abuse, control plane tampering, and stealthy persistence.
4. **Automate triage and enrichment:** Integrate cloud alerts with SOAR workflows to automatically enrich with asset metadata, IAM relationships, and threat intelligence.
5. **Test and tune playbooks:** Run attack simulations, such as token theft and rogue administrator, and refine response procedures based on time to contain and clarity of root cause.

Success Indicators

- ✓ **100% coverage of native cloud audit logs**
  All critical services across AWS, Azure, and GCP forward logs—including control plane, data access, and activity logs—to a centralized platform with long-term retention and searchability.
- ✓ **≥ 90% of logs normalized to a common schema**
  Ingested logs conform to a unified schema that supports correlation across services, identity systems, and threat vectors. Ad hoc parsing is no longer required for investigations.
- ✓ **High-fidelity detections enriched with posture and identity context**
  Detection logic combines telemetry with contextual signals (e.g., misconfiguration states, IAM relationships, exposure levels) to surface meaningful alerts, not raw activity.
- ✓ **Detection-to-containment time improves quarter over quarter**
  Key metrics, such as mean time to detect (MTTD) and mean time to respond (MTTR), are measured, tracked, and reduced through iterative tuning and playbook refinement.
- ✓ **At least one live simulation run per quarter**
  Security teams execute cloud-native attack scenarios at regular intervals. Postmortems identify detection gaps and procedural friction.
- ✓ **≥ 80% of alerts auto-enriched with contextual data**
  Alerts ingested into the SOC include metadata such as associated identities, affected assets, historical access patterns, and exposure risk. Manual enrichment becomes the exception.
- ✓ **Automated response actions preconfigured for top threat scenarios**
  For common attack types, SOAR integrations can autodisable accounts, quarantine workloads, or escalate with preapproved actions.
- ✓ **Incident reviews drive detection engineering updates**
  After every major alert or incident, teams update detection rules and playbooks based on observed TTPs, missed signals, or investigative complexity. No critical learning goes unaddressed.

Software Supply Chain and CI/CD Security

Modern applications rarely run on code written in-house. They depend on open-source libraries, third-party packages, automation scripts, and infrastructure as code—all moving at the speed of DevOps. Each link in the chain carries risk. Attackers know that corrupting the pipeline can bypass even the strongest perimeter. Security must anchor the supply chain from commit to deployment.

What to Evaluate

Segmentation and Exposure Control

CI/CD pipelines introduce scale, speed, and complexity into development workflows. Without controls at build time, vulnerabilities can slip into production through misconfigured infrastructure, outdated dependencies, or malicious inserts. Security engineers must treat pipelines as critical infrastructure, hardening them against misuse and embedding validation at every stage.

- Do we scan all code using both static and dynamic analysis before it's deployed?
- Are security tools integrated directly into CI/CD pipelines for automated feedback and enforcement?
- Have we defined dependency policies that block vulnerable packages or unapproved licenses?
- Can we detect and alert on changes to pipeline configurations or build scripts?
- Are unsigned or unexpected artifacts ever permitted into release stages?

SBOM and Artifact Provenance

Without a reliable software bill of materials (SBOM), you can't prove what's in your code—or where it came from. Attacks on open-source repositories, typosquatting, and dependency confusion have made software provenance a security obligation. Teams must validate the origin, integrity, and trust level of every component shipped to production.

- Are SBOMs generated automatically for every build? Where are they stored?
- Do we validate digital signatures for all third-party libraries and packages?
- Can we trace any deployed artifact back to a specific commit, build job, and set of inputs?
- Are we scanning SBOMs for known vulnerabilities as part of release gating?
- Do we maintain a policy around acceptable software origins and signature schemes?

Control Maturity Grid — Software Supply Chain and CI/CD Security

Capability	Level 0: Absent	Level 1: Ad Hoc	Level 2: Standardized	Level 3: Integrated and Preventive
CI/CD Pipeline Hardening	No pipeline-specific controls exist	Some manual hardening of build systems	Common build systems hardened with shared baselines	Pipelines treat security controls as build-breaking failures
Automated Dependency Scanning	Dependencies are unscanned and unmanaged	Developers run scanners locally or optionally in CI	Dependency scanning enforced in CI/CD pipelines	Dependencies scanned for CVEs and license risk pre-merge
Pipeline Tamper Detection	No monitoring of pipeline changes or config drift	Pipeline changes reviewed manually without alerting	Pipeline changes logged and flagged on merge	Pipeline config and secrets are version-controlled and monitored
SBOM Generation and Enforcement	No SBOM generated or retained	SBOM created sporadically or outside of CI	SBOM generated automatically in every build	SBOMs validated and used for release gating
Artifact Provenance and Integrity	No tracking or validation of artifact origin	Basic tracking of artifact versions, no validation	Artifact metadata includes commit and build trace	Artifacts signed, traceable, and verified before promotion
Policy-Governed Package Usage	No rules on allowed packages or license types	Advisory-only lists of approved or banned packages	Known-vulnerable packages blocked via policy	Dependency policies enforced via automation, not exception

Action Items

1. **Integrate scanning into pipelines:** Apply SAST, DAST, and IaC scanning to every pull request and build job. Fail builds on high-severity findings or policy violations.
2. **Control dependency intake:** Define and enforce dependency policies. Block known vulnerable packages, unverified sources, or noncompliant licenses using dependency scanning tools.
3. **Monitor CI/CD for tampering:** Track changes to pipeline configurations, credential usage, and unusual build activity. Alert on unapproved modifications.
4. **Generate and retain SBOMs:** Use automated tooling to produce SBOMs for every release. Store them alongside build artifacts with immutable metadata.
5. **Enforce signature validation:** Require all critical components, especially external packages, to be signed with a verified key. Block unsigned artifacts from reaching production.

Success Indicators

- **100% of builds produce and retain SBOMs**
  Every release includes a machine-readable software bill of materials stored alongside artifacts. Security teams can query SBOMs by artifact, build, or time frame.
- **≥ 95% of dependencies sourced from approved repositories**
  Third-party packages come from vetted registries with verified maintainers. Dependency scanners enforce source restrictions and block unauthorized intake.
- **All CI/CD pipelines enforce policy-based gating**
  Pipelines fail builds on policy violations (e.g., critical SAST findings, unapproved licenses, unsigned artifacts) before deployment occurs. Enforcement metrics are tracked per pipeline.
- **≥ 90% of artifacts signed and verified predeployment**
  Release components pass signature verification checks as part of automated pipeline steps. Signature enforcement is mandatory for all production workloads.
- **Every deployed artifact is traceable to its origin**
  Security and engineering teams can map each artifact to its originating code commit, associated build job, and dependency graph. Provenance is logged and queryable.
- **No critical unsigned artifacts in production**
  Scans detect no presence of unsigned or unverifiable components in live environments. Exceptions are documented and subject to remediation timelines.
- **Pipeline integrity checks run continuously**
  Monitoring detects any configuration drift, permission elevation, or credential anomaly in CI/CD systems. Alerts trigger immediate review for suspicious pipeline changes.
- **Release approvals require verified inputs and audit trail**
  Release managers approve only those artifacts that have passed all validation gates, including SBOM inspection, vulnerability scanning, and signature enforcement. Every approval has a corresponding audit log.

Governance, Risk, and Compliance

Security doesn't scale without governance. Cloud environments evolve faster than manual oversight can track, and regulatory expectations demand verifiable controls. Governance frameworks align technical controls with business risk, while automation ensures these controls operate continuously.

What to Evaluate

Policy Enforcement and Automation

Cloud misconfigurations result from drift, speed, and inconsistency. Policies must be written as code, versioned like code, and enforced like code. Guardrails must exist at the point of creation, not post hoc audit.

- Have we defined baseline policies across all accounts and regions for resource configurations, encryption, and access?
- Do we enforce tagging requirements for asset classification, ownership, and cost allocation?
- Are policy violations detected and remediated automatically, or do they rely on manual review?
- Can we identify noncompliant resources created via automation or third-party templates?
- Are policies enforced predeployment (e.g., in CI/CD or Terraform plan phases), not just in runtime?

Compliance Mapping

Teams must explicitly map cloud services controls to technical implementations and track their effectiveness across environments. Compliance is achieved through telemetry-backed evidence and measured accountability.

- Do we maintain a centralized control matrix that maps cloud configurations to compliance frameworks?
- Can we generate audit-ready evidence showing enforcement and remediation timelines?
- How frequently do we review control effectiveness across inherited, managed, and custom configurations?
- Are there gaps between security posture and regulatory expectations that remain unresolved?
- Do we treat compliance automation as an engineering problem, versus a checklist exercise?

Control Maturity Grid — Governance, Risk, and Compliance

Capability	Baseline Practices	Improving Practices	Mature Practices
Policy Definition	Policies are documented but inconsistently applied across accounts or services.	Policies are standardized and tracked via configuration templates.	Policies are codified, version-controlled, and enforced via policy as code across all environments.
Automated Enforcement	Violations are detected manually or via ad hoc scripts.	Basic autoremediation is in place for common policy breaches.	Enforcement actions are fully automated, context-aware, and integrated into CI/CD and runtime.
Asset Tagging	Tagging is optional and inconsistently applied.	Required tagging for critical assets is in place, but gaps remain.	All assets are tagged with ownership, data classification, and environment metadata at creation.
Compliance Mapping	Control mappings to frameworks are ad hoc or limited to audits.	A centralized matrix maps technical controls to multiple frameworks.	Compliance mappings are continuously updated, evidence is autocollected and mapped to policy state.
Audit Evidence Collection	Evidence is gathered during audit cycles and often manually compiled.	Evidence is collected periodically through logging and basic snapshots.	Evidence collection is automated and real-time, with dashboards showing enforcement and resolution.
Remediation Tracking	Noncompliance is tracked via tickets or static reports.	Remediation timelines are logged and reviewed monthly.	Control violations trigger SLAs and are tracked through automated resolution workflows and metrics.
Control Effectiveness Review	Reviews occur post-incident or annually.	Reviews occur quarterly and inform updates to baseline configurations.	Effectiveness is monitored continuously via posture telemetry, control drift indicators, and audits

Action Items

1. **Codify policy as code:** Use native CSPM policies to define and enforce configuration standards across IaC, APIs, and consoles.
2. **Enforce asset tagging:** Require metadata on every deployed asset (e.g., environment, owner, data classification, business unit). Reject untagged or misclassified resources automatically.
3. **Remediate noncompliance automatically:** Configure workflows to alert, quarantine, or autocorrect resources that drift from policy. Track resolution time per control.
4. **Map controls to frameworks:** Build or adopt a control matrix that ties cloud configurations to compliance requirements. Use that matrix to drive audit evidence collection.
5. **Instrument evidence collection:** Automate collection of logs, config snapshots, and control state to support internal and external audit cycles without disruption.

Success Indicators

- ✓ **≥ 95% of resources deployed with policy-as-code enforcement**
  New infrastructure passes through predeployment checks where policy-as-code engines validate configuration, tagging, and access controls before provisioning.
- ✓ **100% of production assets tagged with ownership and classification metadata**
  Every production resource includes metadata for owner, environment, and data sensitivity. Dashboards reflect tagging compliance with real-time status and exception tracking.
- ✓ **Noncompliant resources remediated within defined SLAs**
  Policy violations trigger alerts, automated corrections, or quarantines. The median time to policy compliance is tracked and reported weekly. SLA breaches are audited quarterly.
- ✓ **≥ 90% policy coverage across required compliance frameworks**
  Control matrices show traceable mappings between cloud configurations and NIST, ISO 27001, SOC 2, and other frameworks. Gaps are flagged and prioritized.
- ✓ **Audit evidence generated automatically and on demand**
  Evidence artifacts (e.g., control states, logs, configuration timelines) are generated without manual collection. Auditors can validate controls without disrupting operations.
- ✓ **Compliance reviews completed quarterly**
  Every quarter, teams review control coverage, posture drift, and audit-readiness metrics. Reports include remediation timelines, unresolved risks, and updated mappings.
- ✓ **Policy enforcement metrics are visible to stakeholders**
  Dashboards report policy violations, exception approvals, and remediation trends. Executive stakeholders use these metrics to track alignment between governance and security posture.

Continuous Exposure Management

Cloud environments never stop changing. New assets appear without notice, configurations drift, and exposures emerge faster than most teams can assess, let alone fix. Continuous exposure management shifts the focus from theoretical risk to what adversaries can actively exploit. It enables organizations to move from passive hardening to dynamic defense.

What to Evaluate

Attack Surface Discovery

Most organizations don't know what they've exposed until someone else finds it. Shadow assets, legacy endpoints, misconfigured APIs, and abandoned development tools all expand the attack surface beyond what's reflected in the asset inventory. Continuous discovery must include external and internal perspectives and extend across every cloud account and region.

- Can we identify every internet-facing asset, including unmanaged and legacy services?
- Are we monitoring ephemeral and autoscaled resources that may expose services unexpectedly?
- Do we include DNS records, load balancers, public IPs, and misconfigured storage buckets in our discovery scope?
- Are we continuously scanning for exposed secrets, credentials, or open management interfaces?
- Can we confirm when an exposed asset was last seen and whether it's been remediated?

Exposure Prioritization

Priority is the problem. Security teams face thousands of findings, but only a fraction present real risk. Exposure prioritization must combine asset criticality, identity relationships, misconfiguration impact, and active threat context to identify what matters now.

- Are we correlating cloud misconfigurations with current threat actor TTPs or exploit activity?
- Can we identify attack paths that chain multiple low-risk exposures into privilege escalation or data exfiltration?
- Do we prioritize exposures based on blast radius, as well as who or what could be impacted?
- Are we aligning exposure triage with business context, such as data classification or service tier?
- Do we have clear criteria to suppress noise and highlight exploit-ready conditions?

Control Maturity Grid — Continuous Exposure Management

Capability	Baseline Practices	Improving Practices	Mature Practices
Attack Surface Visibility	Teams rely on asset inventories without external scanning or validation.	Internet-facing assets are scanned periodically using internal tools.	Internal and external scanners run continuously across all accounts, regions, and resource types.
Shadow Asset Detection	Shadow assets are found manually or only after external disclosure.	Limited automation discovers assets outside approved inventories.	Shadow assets—including legacy endpoints, DNS records, and dev tools—are surfaced and triaged in real time.
Exposure Prioritization	Alerts are triaged by severity alone, without blast radius or business context.	Exposures are scored with some context around asset value and potential impact.	Prioritization incorporates business criticality, exploit signals, identity mapping, and lateral movement potential.
Attack Path Mapping	No visibility into how exposures can be chained across identities or systems.	Attack paths are occasionally modeled post-incident or by red teams.	Graph-based exposure modeling is integrated into security tooling to visualize real attack paths.
Threat Intelligence Correlation	Misconfigurations and vulnerabilities are reviewed in isolation.	Teams enrich select findings with external threat data.	Exposure data is continuously correlated with active campaigns, scanning activity, and known adversary TTPs.
Remediation Readiness	Fixes rely on manual triage and ticketing across fragmented teams.	Some automated alerts lead to preapproved remediation steps.	High-priority exposures trigger contextual, automated responses with clear SLAs and audit trails.

Action Items

1. **Discover exposed assets continuously:** Deploy external and internal scanners to detect internet-facing assets, exposed management ports, open storage, and unauthorized APIs. Include DNS, CDNs, and load balancer frontends.
2. **Run posture assessments regularly:** Automate posture evaluations to detect misconfigured IAM, unencrypted services, open access policies, and untagged resources. Feed findings into a unified exposure view.
3. **Correlate with threat intelligence:** Integrate detection with external intelligence sources to identify exploit attempts or malware targeting similar configurations.
4. **Visualize attack paths:** Use graph-based exposure mapping to identify lateral movement opportunities, chained misconfigurations, and accessible privilege escalation routes.
5. **Focus remediation on exploit-ready risk:** Triage exposures not just by severity but by exploitability, blast radius, and business criticality. Assign SLAs based on actual versus theoretical risk.

Success Indicators

- ✓ **100% visibility into internet-facing assets across all cloud accounts**
  Asset discovery tools inventory every public IP, DNS record, and externally exposed service and is updated in near real time and validated against provider APIs and threat intelligence sources.
- ✓ **All ephemeral assets logged within 60 seconds of creation**
  Autoscaled, short-lived, and dynamic workloads are automatically registered in asset management systems. Telemetry confirms visibility within 1 minute of spin-up.
- ✓ **≥ 95% of findings enriched with exploitability or threat context**
  Exposure data includes correlated indicators (e.g., Shodan indexing, scanning behavior, known TTP matches, access potential). Prioritization is based on evidence, not static risk scores.
- ✓ **Exposure triage reflects business and blast radius context**
  Risk scores incorporate asset sensitivity, privilege level, and potential impact. Internal dashboards tie exposure to real-world consequences.
- ✓ **Attack path modeling completed and refreshed weekly**
  Exposure graphs identify privilege escalation routes and lateral movement paths. Models update automatically and inform weekly triage workflows.
- ✓ **Mean time to remediate active exposures under 72 hours**
  Exploit-ready exposures, confirmed by telemetry or intelligence, trigger workflows with defined SLAs. Tracking shows the median time to closure under three days.
- ✓ **Noise suppression rate ≥ 85%**
  Alerting systems suppress nonexploitable misconfigurations. Only high-fidelity, actionable exposures are routed to remediation pipelines. Suppression criteria are tested and tuned monthly.

AI Risk Governance

AI systems enter cloud environments through APIs, SDKs, orchestration pipelines, and embedded inference. They expose new surface areas, both in architecture and behavior, that traditional controls don't cover. Governance must address the full AI lifecycle, from training data and prompt inputs to model exposure and audit transparency.

What to Evaluate

- Do we maintain a complete inventory of deployed AI/ML services, their access paths, and dependencies?
- Can we track data lineage and verify input sources for training and inference?
- Have we implemented runtime monitoring for misuse, prompt injection, or unexpected output behaviors?
- Do we review access controls, authentication mechanisms, and permission boundaries for model endpoints?
- Are we testing model behavior under adversarial input conditions, drift scenarios, and edge cases?

Control Maturity Grid — AI Risk Governance

Capability	Ad Hoc	Defined	Measured	Enforced
Model Inventory	Incomplete or informal	All models listed, not versioned	Inventory tied to CI/CD and API discovery	Version-controlled, reviewed quarterly
Input Governance	No sanitization	Static filters applied	Input validation updated from abuse patterns	Input pipelines tested, blocked on risk
Prompt Injection Monitoring	Not monitored	Known bad patterns logged	Detection rules tuned by model type	Prompt tampering triggers alerts, isolation
Adversarial Testing	Not performed	Simulated abuse cases in backlog	Synthetic inputs run prerelease	Red team model abuse simulation enforced
AI-SPM Coverage	Not defined	AI components added to posture scans	AI telemetry included in compliance checks	Full lifecycle governance enforced via policy

Action Items

1. **Inventory AI systems:** Document every deployed, tested, or retired model. Include training dataset summaries, model type, inference interface, and associated APIs.
2. **Apply model classification and risk labeling:** Tag each model based on input sensitivity, output exposure, external dependencies, and compliance relevance.
3. **Enforce prompt and input controls:** Define input sanitization rules, content restrictions, and rate limits. Monitor for known prompt injection or misuse patterns.
4. **Instrument adversarial testing workflows:** Create synthetic inputs that simulate abuse attempts, drift, or manipulation. Run them against all high-impact models quarterly.
5. **Log all inference activity:** Ensure requests and outputs are auditable. Preserve logs for forensic analysis and behavioral baselining.
6. **Integrate AI into policy as code:** Treat model behavior and access as governed infrastructure. Apply continuous validation checks and surface violations to security workflows.

Success Indicators

- ✓ **All production models mapped and classified by sensitivity**
  Each active model includes ownership, data sources, intended use, and exposure level. Classification appears in audit logs and governance dashboards.
- ✓ **≥ 95% of AI systems tested for adversarial behavior quarterly**
  Test coverage spans prompt manipulation, output drift, and inference abuse. Failures trigger model updates or policy revision.
- ✓ **Prompt misuse detection in place for ≥ 90% of exposed endpoints**
  Every public or semipublic interface includes rules to detect token misuse, embedded exploits, or abusive chaining.
- ✓ **AI policies enforced through CI/CD and runtime instrumentation**
  Policy violations surface during pipeline execution or model startup.
- ✓ **Governance state visible on demand**
  Security and compliance teams can instantly view model posture, enforcement coverage, last test result, and open findings by asset.

Cloud Security Checklist Index

CORTEX® CLOUD
BY PALO ALTO NETWORKS

The Code to Cloud to SOC Advantage

Cortex Cloud unifies code, infrastructure, and runtime telemetry within a single data lake, where signals from the supply chain to runtime are traced and correlated. The solution builds an uninterrupted map of how risk originates, propagates, and threatens critical systems.

Its unified architecture powers AI-driven detection and automates response with full context. Findings are scored based on real exposure and business impact and grouped into cases that reflect the complete story—misconfigurations linked to identity risk, code flaws tied to active exploit paths, runtime anomalies enriched with threat intelligence.

Security teams work smarter and move faster. For the SOC, that means quick triage and clear response paths. For cloud and AppSec teams, it means prioritization that informs decisions and expedites action. Organizations achieve a unified defense posture that scales with the environment and adapts in real time.

Ready for continuous posture assessment, intelligent automation, and high-fidelity detection across the attack surface?

Discover how a unified security strategy can empower your organization to outpace modern threats.

Schedule a demo of Cortex Cloud today

3000 Tannery Way
Santa Clara, CA 95054

Main: +1.408.753.4000
Sales: +1.866.320.4788
Support: +1.866.898.9087