SANS | Research Program

BUYERS GUIDE



Microsegmentation Buyers Guide:

How to Evaluate Solutions for Zero Trust and Hybrid Cloud

Written by Dave Shackleford,
SANS Senior Instructor

December 2026

zscaler™

©2026 SANS™ Institute

Introduction: Why Microsegmentation Matters

Enterprise applications no longer reside inside one predictable network boundary. They span data centers, multiple public clouds, containers, and services connected by machine identities and APIs. That distribution creates legitimate east-west communication, but it also gives attackers pathways to move from an initial foothold into critical systems. Traditional perimeter firewalls remain useful, yet they rarely describe which specific workloads should be allowed to talk to one another.

Microsegmentation applies fine-grained controls to workload and application communications. The objective is not to block every connection or to replace every existing firewall. It is to make unnecessary connections unavailable, reduce lateral movement, and limit the number of systems an attacker can reach after compromise. Done well, segmentation becomes a repeatable part of application deployment and change management rather than a collection of one-off firewall projects.

This guide explains what to evaluate, how to test competing architectures, and which operational characteristics matter after initial deployment. It treats vendor feature claims as hypotheses to verify in the buyer's own environment, especially when functionality varies by operating system, cloud, Kubernetes distribution, or subscription tier.

Microsegmentation in a Zero Trust Architecture

Macrosegmentation separates broad network zones, virtual private clouds (VPCs), virtual networks (VNETs), or business domains. Microsegmentation narrows access further—to workload groups, application tiers, processes, pods, or services—so that being on an approved network does not imply permission to reach every resource there. Both layers remain relevant. Network firewalls and cloud-native controls can define major trust boundaries, while local or workload-aware controls constrain movement inside them.

Zero trust principles translate into explicit communication requirements, narrowly scoped permissions, continuous context, and the assumption that any one workload or credential could be compromised. Segmentation addresses network reachability; it does not replace strong authentication, privileged access management, endpoint detection, application authorization, or data protection. An application that is allowed to reach a database still requires appropriate identity and query-level controls.

The desired end state is simple to state but challenging to implement: An application can reach its documented dependencies, and nothing else, regardless of whether components run on premises or in the cloud. Therefore, buyers should evaluate not only blocking capabilities but also how a product discovers dependencies, models policy, handles changes, and explains failures.

Start with Business and Security Outcomes

A successful program should first reduce exposure around critical services. Chasing a theoretically perfect least-privilege ruleset across every workload can delay meaningful protection for years. A more useful starting point is to identify high-value applications, shared administrative infrastructure, sensitive data stores, and pathways that enable ransomware propagation or credential-based lateral movement.

Translate that priority into measurable outcomes: fewer reachable workloads from a compromised server, higher coverage of critical applications under enforced policy, faster discovery and assignment of new workloads, fewer broad exceptions, and lower effort to approve and troubleshoot changes. Regulatory and audit evidence is a useful secondary benefit, but it should follow demonstrated control effectiveness rather than substitute for it.

Agree on ownership before selecting technology. Application teams know legitimate dependencies, network and platform teams own connectivity and provisioning, security architects define trust boundaries, and the SOC needs actionable events. Without these relationships, even an excellent technical platform can produce a brittle policy program.

Core Buyer Evaluation Categories

The following 16 categories form the technical and operational evaluation framework. Each category includes a practical test that can be performed during a proof of concept (POC). Capabilities should be assessed for the exact operating systems, deployment models, cloud services, and licenses in scope.

1. Asset Discovery, Inventory, and Coverage

Inventory is not synonymous with coverage. A discovery integration may list cloud instances while providing no traffic telemetry or enforcement for those same assets. Buyers should distinguish assets found, assets classified, assets observed, and assets actually protected. That distinction matters most for unmanaged servers, ephemeral resources, legacy operating systems, and workloads deployed outside normal provisioning processes.

A mature solution should be evaluated as follows:

- Automatically discover workloads across on-premises, colocation, AWS, Azure, GCP, and Kubernetes.
- Inventory managed and unmanaged resources; identify blind spots before enforcement.
- Capture cloud metadata, tags/labels, VPC/VNET, subnet, IP, hostname, namespace, pod, node, and application context.
- Evaluate the ability to capture real-time workload metadata, including deep integrations with cloud provider services like Cloud Instance Metadata Service (IMDS) and other native cloud services.
- Assess whether the discovery and inventory data is presented in a unified console that supports both macrosegmentation and microsegmentation views, and facilitates the delivery of policy recommendations.
- Measure discovery freshness and the time required for newly created workloads to appear.
- Assess coverage for VMs, bare metal, containers/Kubernetes, ephemeral workloads, and serverless where applicable.
- Evaluate configuration management database (CMDB)/cloud inventory synchronization and reconciliation.

BUYER VALIDATION:

Demonstrate onboarding a new cloud instance and a previously unknown server. Record time to discovery, the metadata available, and whether the console accurately labels each as monitored, enforceable, or unsupported.

---

2. Traffic Flow Visibility and Application Dependency Mapping

Application dependency maps are only useful when they help owners separate necessary flows from historical noise. Infrequent backup jobs, patch windows, certificate renewal, directory lookups, and failover paths are easy to miss in a short observation period. A platform should expose the evidence behind each flow and make it possible to compare current behavior with an earlier baseline.

A mature solution should be evaluated as follows:

- Observe workload-to-workload and process/service communications in near real time.
- Map initiators, receivers, ports, protocols, processes, and application dependencies.
- Visualize flows at host-to-host, group-to-group, application-tier, and cross-cloud levels.
- Support historical flow analysis to account for infrequent jobs, batch processes, backup, patching, and failover paths.
- Export or stream flow telemetry for SIEM, data lake, NDR, and troubleshooting workflows.

BUYER VALIDATION:

Select an application with a scheduled batch job and an external dependency. Verify source, destination, direction, service, timing, and process context where available; then test whether the dependency survives a proposed restrictive policy.

---

3. Workload Identity, Context, and Dynamic Grouping

Static IP rules tend to break as cloud instances are replaced and applications scale. Contextual grouping can reduce this churn, but only if tags, ownership metadata, and labels are reliable. The buyer should define who governs these attributes, how conflicting tags are resolved, and what happens when metadata is missing or stale.

A mature solution should be evaluated as follows:

- Move beyond IP-only policy by using workload identity and contextual attributes.
- Create dynamic groups from cloud tags, labels, application role, environment, ownership, location, namespace, or other metadata.
- Automatically update membership as workloads scale, move, or are redeployed.
- Evaluate resilience to poor or inconsistent tagging and the ability to combine multiple attributes.
- Determine whether common policy objects can be reused across segmentation and adjacent security controls.

BUYER VALIDATION:

Change a workload IP and application tag, then observe group membership and effective policy. Intentionally introduce an invalid tag and determine whether the system fails safely and alerts an owner.

---

4. Policy Modeling and Least-Privilege Design

Least privilege is an operational process rather than a one-time rule-generation exercise. Teams need a policy model application owners can understand, security teams can review, and operators can troubleshoot. Explicitly define the treatment of shared services, administrative protocols, cross-environment dependencies, exceptions, and emergency access.

A mature solution should be evaluated as follows:

- Support intuitive source-to-destination/service policy models without requiring deep network redesign.
- Model allowlists for required application dependencies and default-deny for unnecessary communication.
- Support host-to-host, group-to-group, and app-tier policy.
- Provide policy templates or reusable service/application objects.
- Evaluate rule ordering, precedence, conflict detection, shadowed rules, exceptions, and policy inheritance.
- Assess whether policy can follow the workload rather than the subnet.

BUYER VALIDATION:

Build a three-tier application policy with explicit DNS, identity, monitoring, and backup dependencies. Check rule precedence and the ability to explain every permitted path.

---

5. AI/ML-Assisted Policy Recommendations

AI-assisted recommendations can shorten analysis, but recommendations are not proof of business necessity. Training windows may include unauthorized traffic, compromised hosts, or obsolete dependencies. Look for traceable supporting flows, understandable grouping logic, administrator approval, and evidence of what the recommendation would block.

A mature solution should be evaluated as follows:

- Analyze observed flows to recommend workload groups and candidate least-privilege rules.
- Identify uncovered assets or communications that fall outside intended segmentation policy.
- Explain why a recommendation exists and which flows support it.
- Provide confidence, impact, and exception context rather than opaque auto-enforcement.
- Require human review/approval and auditable change history for AI-generated policy.

BUYER VALIDATION:

Generate recommendations from a mixed dataset containing legitimate traffic. Have an administrator explain, reject, and modify the recommendation before enforcement.

---

6. Enforcement Architecture and Granularity

The enforcement point determines both the protection boundary and the failure modes. Host controls can stop local lateral movement without changing the network path, while gateways and network controls may provide broader protocol inspection at the cost of routing dependencies. Claims about Layer 7 security, inline DLP, or threat prevention must be tested independently from basic segmentation. Visibility into a process or fully qualified domain name (FQDN) does not automatically mean that attribute can be used in an enforcement rule.

A mature solution should be evaluated as follows:

- Understand exactly where enforcement occurs: host, kernel, hypervisor, network, proxy/gateway, cloud control, or combinations.
- Validate that these converged security controls can be applied without requiring separate agents or complex traffic routing, distinguishing a unified platform from a collection of point products.
- Evaluate OS-native enforcement mechanisms and their behavior under agent/service failure.
- Support inbound, outbound, and internal controls where required.
- Assess supported enforcement for TCP/UDP, IPv4/IPv6, ports, and service definitions; distinguish visibility metadata from actual policy enforcement capabilities.
- Validate local enforcement without unnecessary traffic hairpinning or cloud egress.
- Evaluate how unauthorized connections are blocked or logged.

BUYER VALIDATION:

Test TCP, UDP, and IPv6 behavior as applicable, including connections from an unauthorized peer. Identify precisely which traffic is blocked, merely observed, or sent to a separate inspection service, and measure any added latency.

---

7. Kubernetes and Cloud-Native Segmentation

Kubernetes presents a distinct challenge because pods are short-lived and IP addresses are disposable. Namespace or label-based visibility does not establish enforceable pod-level isolation. Buyers should separate current support from planned functionality and should validate the effect of autoscaling and rolling deployments.

A mature solution should be evaluated as follows:

- Discover clusters, nodes, pods, namespaces, labels, services, and relevant runtime context.
- Apply policy at a level that survives pod churn and autoscaling.
- Test visibility and east-west controls within a cluster, across clusters, and between Kubernetes and VM/database tiers.
- Assess deployment methodology and operational implications.

BUYER VALIDATION:

Deploy and replace pods during the POC; verify the policy remains attached to the intended service. Test same-namespace, cross-namespace, cross-cluster, and VM-to-pod paths only where those capabilities are supported.

---

8. Hybrid and Multicloud Consistency

A unified console is valuable only if it actually reduces policy translation and operational differences. AWS security groups, Azure NSGs, GCP firewall controls, and host agents have different semantics and limitations. Buyers should understand whether the platform enforces a common intent natively, orchestrates multiple provider-specific controls, or simply displays them together.

A mature solution should be evaluated as follows:

- Use a common policy model across data center, colocation, AWS, Azure, GCP, and cloud regions.
- Protect cloud-to-cloud, cloud-to-data-center, region-to-region, and VPC/VNET-to-VPC/VNET communication.
- Avoid separate policy languages and management planes for each cloud wherever possible.
- Evaluate how cloud metadata differences are normalized into a common policy abstraction.
- Assess support for migrations where application components temporarily span environments.

BUYER VALIDATION:

Evaluate macro and micro flows together and apply enforcement capabilities appropriate for macrosegmentation (L7, proxy/tunneling) vs. those appropriate for microsegmentation (L3/L4). Document which rules, groups, logs, and exceptions are preserved and which require manual translation or redeployment.

---

9. Zero Trust Platform and Adjacent Control Integration

Segmentation should complement existing security controls rather than create ambiguous ownership. An NGFW may inspect application traffic, a cloud firewall may control subnet boundaries, and a host agent may restrict workload-to-workload connections. Buyers need an explicit architecture showing where each decision is made, which system is authoritative, and how conflicting controls are diagnosed.

A mature solution should be evaluated as follows:

- Evaluate integration between workload microsegmentation and user-to-application zero trust access.
- Determine whether workload identity/context can be reused by SWG, firewall, DLP, TLS inspection, ZTNA, or other controls.
- Assess interoperability with existing NGFW, cloud firewall, EDR/CWPP, CNAPP, IAM, PAM, and service-mesh investments.
- Avoid duplicated enforcement and inspection that add latency or operational ambiguity.
- Define which control plane owns macrosegmentation vs. microsegmentation.

BUYER VALIDATION:

Trace an allowed and a denied flow through ZTNA, network firewall, cloud controls, and host enforcement. Verify that logs identify the actual decision point and that duplicate inspection is intentional.

---

10. Policy Simulation, Staging, and Safe Rollout

Visibility-only mode is the safest starting point, but it should lead to a measured enforcement decision. Simulation should account for both common traffic and rare recovery workflows. Buyers should demand an emergency bypass process with a named owner, expiration, audit trail, and reliable rollback behavior.

A mature solution should be evaluated as follows:

- Provide observe/visibility-only modes before blocking.
- Simulate proposed deny rules against historical and current flows.
- Show which applications, hosts, processes, or business services would be affected before enforcement.
- Support staged rollout by application, environment, resource group, or business unit.
- Provide rapid rollback, emergency bypass, and time-bound exception mechanisms.
- Test failover, disaster recovery, maintenance, backup, and rare application workflows before default-deny.

BUYER VALIDATION:

Run a simulated deny against an application dependency, activate the rule in a test segment, and demonstrate rollback while measuring elapsed time and residual access.

---

11. Change Management, Automation, and Policy as Code

Automation is most useful when it preserves review and traceability. Buyers should verify which APIs and integrations are shipping now, their rate limits, authentication model, error handling, and whether they cover the same objects available in the GUI. “Policy as code” should not be assumed simply because an API exists.

A mature solution should be evaluated as follows:

- Expose APIs and automation interfaces for inventory, policy, reporting, and life cycle operations.
- Assess available APIs and supported integrations with infrastructure automation and deployment workflows; distinguish shipping integrations from roadmap commitments.
- Support versioning, approvals, peer review, rollback, and separation of duties.
- Evaluate how changes to workload metadata or group membership affect applied segmentation policy and how administrators can investigate discrepancies.
- Assess whether policy changes can be tested and promoted consistently from development to production.

BUYER VALIDATION:

Create, review, deploy, and roll back a policy change through the supported interface. Deliberately alter a workload tag and determine how the platform reports changes to effective access.

---

12. Agent Life Cycle, Resilience, and Tamper Protection

Agent design affects the viability of broad deployment. Operating system and kernel support, EDR coexistence, upgrade scheduling, and behavior during cloud disconnection must be understood before production. A clear distinction is needed between loss of management connectivity and failure of the local enforcement component.

A mature solution should be evaluated as follows:

- Evaluate agent footprint, CPU/memory overhead, upgrade model, and compatibility with supported operating systems.
- Support centralized rollout through enterprise software/configuration management.
- Document supported enforcement behavior during agent disconnection or control-plane outages, and test observable effects on existing and new connections.
- Protect enforcement components from unauthorized stop/kill/tampering.
- Support staged agent upgrades (pilot group before broad deployment), maintenance scheduling, clear error reporting, and visibility into installed agent versions.
- Document kernel/OS dependencies and interactions with EDR, host firewall, and hardening controls.

BUYER VALIDATION:

Test stopping the agent process when anti-tamper is enabled. Also consider running a staged upgrade on a pilot group and confirm version reporting and rollback options.

---

13. Performance, Scale, and Availability

Performance must be measured under the buyer's actual workload and inspection mix, not just a published peak throughput figure. Important dimensions include connection rate, policy count, agent CPU overhead, latency-sensitive applications, metadata churn, and time for a new workload to receive the correct policy.

A mature solution should be evaluated as follows:

- Establish tested limits for workloads, agents, groups, policies, flows, events, and concurrent changes.
- Measure policy propagation and convergence time during autoscaling and cloud churn.
- Quantify latency and throughput impact for enforcement paths.
- Validate availability and enforcement behavior during controller/service outages.
- Assess regional/global architecture, data residency, and management-plane dependencies.

BUYER VALIDATION:

Test at representative concurrency and change rates. Measure latency, CPU and memory, propagation delay, and service continuity during controller or connectivity loss.

---

14. Monitoring, Troubleshooting, and Security Operations

Segmentation often becomes a day-two troubleshooting problem: A legitimate application fails, and teams must quickly identify which policy blocked it. Effective tooling links the source and destination to workload identity, application owner, rule, action, and supporting evidence. SIEM export should preserve this context without forcing the organization to ingest every allowed flow.

A mature solution should be evaluated as follows:

- Provide searchable flow logs, policy decisions, blocked/allowed events, and effective-policy views.
- Answer the question, “Why was this flow blocked/allowed?” quickly enough for operations teams.
- Correlate workload metadata, policy, process/service, and traffic evidence in one investigation workflow.
- Integrate alerts and telemetry with SIEM/SOAR and incident-response processes.
- Support packet/flow evidence or equivalent troubleshooting data where appropriate.
- Measure mean time to diagnose segmentation-related application failures.

BUYER VALIDATION:

Intentionally block a valid dependency and time the investigation from help-desk ticket to root cause. Export representative blocked-flow and administrative events to the SIEM and confirm field completeness.

---

15. Governance, RBAC, Audit, and Compliance

Governance needs more than an annual report. Large enterprises require delegated administration, separation of duties, exception approval, and a reliable record of policy changes. Evidence should demonstrate what was protected at a point in time, who approved an exception, and whether the exception expired as planned.

A mature solution should be evaluated as follows:

- Provide granular administrative RBAC and separation of policy authoring, approval, and operations.
- Maintain immutable/auditable records of policy and administrative changes.
- Define export options for audit and log data.

BUYER VALIDATION:

Produce a report of critical applications, enforcement status, violations, and overdue exceptions.

---

16. Operational Simplicity, Licensing, and TCO

The cost of segmentation is often dominated by operational effort rather than the initial subscription. Compare the number of consoles, deployment work, tagging cleanup, change tickets, support responsibilities, and log ingestion. Buyers should model a realistic three-year rollout, including assets that need visibility but cannot support an agent or enforcement.

A mature solution should be evaluated as follows:

- Compare deployment effort, infrastructure requirements, management planes, and staffing overhead.
- Understand pricing units: workload, host, core, flow/traffic, cloud consumption, feature tier, or flat subscription.
- Identify separately licensed visibility, recommendation, enforcement, Kubernetes, reporting, or integration capabilities.
- Model three-year costs including agents, gateways/appliances, cloud egress, logging, professional services, and operations.
- Assess whether consolidation genuinely retires point products or simply adds another control layer.
- Evaluate time-to-value: discovery, first policy, first protected application, and expansion to enterprise scale.

BUYER VALIDATION:

Price a pilot, a one-year expansion, and enterprise scale using identical workload assumptions. Include professional services, cloud egress, storage, SIEM costs, exception management, and staff hours.

Architecture Patterns Buyers Should Compare

There is no universal enforcement architecture. Host-based segmentation generally provides local context and can reduce the need for traffic hairpinning, but it also requires operating-system compatibility and agent life cycle management. Network- or firewall-based approaches are familiar and may work well for unmanaged assets, yet are frequently coarser and tied to topology. Cloud-native security groups and firewall rules fit provider-native operations but can create inconsistent policies across clouds.

Kubernetes-native approaches can use NetworkPolicy, Container Network Interface (CNI) implementations, and service-mesh constructs to express pod or service boundaries, but they do not automatically extend to conventional servers. Proxy and zero trust connectivity designs can reduce exposed network reachability, especially for selected application flows, although they should not be assumed to provide complete local east-west isolation. Many mature organizations use a hybrid architecture: macrosegmentation at network boundaries plus fine-grained enforcement where workload context is available.

Pattern	Primary Strength	Principal Trade-off
Host-based	Local workload context and fine-grained controls	Agent coverage and compatibility
Network/firewall	Established controls and shared inspection	Topology dependence and coarse policy with no protection between hosts within a subnet
Cloud-native	Provider integration and automation	Cross-cloud inconsistency
Kubernetes-native	Pod/service-level constructs	Limited reach outside clusters
Proxy/zero trust connectivity	Reduced broad reachability	Coverage and path constraints
Hybrid	Layered protection and flexibility	Ownership and troubleshooting complexity

Proof of Concept: Test What Will Happen in Production

A meaningful POC should prove that the solution can safely move from observation to enforcement, survive change, and provide evidence when something goes wrong. Select at least one conventional three-tier application, one cloud-hosted application, and one critical or regulated service. Assess Kubernetes visibility separately from enforcement when supported functionality differs.

1. **Baseline:** Observe representative traffic over a window that includes scheduled jobs, backups, maintenance, and known failover paths.
2. **Discover and group:** Inventory workloads, validate owners, and test how IP, tag, and instance changes affect dynamic membership.
3. **Recommend:** Generate candidate rules from observed traffic and require a knowledgeable administrator to validate AI-assisted suggestions.
4. **Simulate:** Predict the impact of a proposed deny rule, document exceptions, and secure application-owner approval.
5. **Enforce:** Apply restricted policy to the test application and confirm that authorized dependencies continue to work.
6. **Attack-path test:** Attempt unauthorized connections between peers, tiers, environments, and administrative services.
7. **Failure test:** Interrupt agent, controller, or cloud connectivity and record behavior for existing and new connections.
8. **Operate:** Investigate an intentional false block, roll back safely, export evidence, and review exception expiration.
9. **Measure:** Record performance overhead, policy propagation, analyst effort, and application-owner participation.

Require each vendor to submit evidence rather than demonstrations alone: policy screenshots or exports, flow logs, timestamps, observed traffic outcomes, supported-version documentation, and a clear statement of any manual workarounds. Mark capabilities as generally available, limited to a particular platform, custom-built, or roadmap only.

Implementation and Operating Model

Phase 1: Inventory and ownership: Identify application criticality, owners, environments, and candidate protection groups.

Phase 2: Observe and map: Capture sufficient traffic history, including uncommon workflows, to avoid accidental outages.

Phase 3: Group and model: Define dynamic identities, shared services, and candidate least-privilege policies.

Phase 4: Simulate and validate: Review changes with application owners and test failover, backup, and maintenance dependencies.

Phase 5: Enforce progressively: Start with high-value applications and clear unauthorized paths; expand in measured waves.

Phase 6: Operationalize: Assign change approval, SOC triage, exception expiration, and emergency rollback ownership.

Phase 7: Optimize: Retire stale rules, improve metadata hygiene, expand automation, and measure attack-path reduction.

Do not mistake a successful first application for a scalable program. The second and third deployment waves reveal whether metadata standards, change approval, support staffing, and troubleshooting workflows can sustain expansion. Capture lessons from each wave and update the policy design standard before moving to more critical applications.

Roles and Governance

Security architecture: Defines segmentation principles, target architecture, and exception guardrails

Cloud/platform engineering: Owns workload metadata, supported deployment patterns, and provisioning integration

Network security: Owns macrosegmentation, routing dependencies, and cross-control troubleshooting

Application owners: Confirm required dependencies and accept operational impact of proposed policy

SOC and incident response: Investigate blocked flows and use segmentation telemetry in containment workflows

GRC and audit: Map evidence to applicable requirements and track exceptions and attestations

Change governance: Controls approvals, emergency changes, rollback, and separation of duties

Common Mistakes and Red Flags

- Enforcing before observing a complete application cycle, including backup and disaster recovery
- Using static IPs and subnets as permanent application identity
- Creating thousands of host rules that reproduce legacy firewall sprawl
- Treating a discovery dashboard as proof that every workload is protected
- Assuming Kubernetes visibility also means enforceable pod-level policy
- Trusting AI-generated recommendations without supporting evidence or human review
- Deploying without a tested emergency rollback and time-limited exception process
- Ignoring agent compatibility, EDR coexistence, and controller failure behavior
- Accepting POCs that show only dashboards instead of actual blocked connections
- Underestimating the full cost of enforcement, logging, and ongoing policy administration

Measuring Program Success

Report program outcomes at the application level, not just the number of agents installed. Metrics should reflect what is understood, what is enforced, what remains reachable, and how much operational effort is required to maintain protection.

Metric	What It Indicates
Critical applications with validated dependency maps	Visibility quality
Critical applications under enforced least-privilege policy	Control coverage
Explicitly allowed vs. broadly permitted communications	Least-privilege maturity
Reduction in reachable workloads or attack paths from a compromised workload	Blast-radius reduction
Time from workload creation to correct policy assignment	Dynamic control
Recommendation-to-enforcement cycle time	Operational agility
Number and age of active exceptions	Governance
Segmentation-related incidents and mean time to resolution	Operational quality
Approved policy changes completed through automation	Automation
Stale or unused rules removed	Policy hygiene
Administrative effort per protected application	Sustainability

For the blast-radius metric, define a repeatable test: Select a representative workload, identify reachable peers before enforcement, repeat the same authorized network probes after enforcement, and calculate the reduction. Pair that result with application availability testing; a large reduction in reachability is not a success if essential workflows fail.

Wrapping Up: Buy for Sustainable Enforcement

Microsegmentation can materially reduce lateral movement and the impact of a compromised workload, but success depends on far more than the ability to block ports. Buyers should favor solutions that provide trustworthy dependency visibility, understandable policy, proven enforcement, safe rollout, and manageable operations across the environments they actually run. Metrics should reflect what is understood, what is enforced, what remains reachable, and how much operational effort is required to maintain protection.

The best choice may be a combination of controls rather than one universal platform. Evaluate where enforcement occurs, which assets are outside its reach, how it coexists with existing investments, and whether a real application team can approve and maintain its policies. The accompanying buyer checklist provides a practical way to capture these findings consistently and compare vendor claims against POC evidence.

Microsegmentation Buyer Evaluation Checklist

Companion worksheet to the Microsegmentation Buyer's Guide | December 2026

For each item, mark Required or Desired based on your organization's priorities, record whether the vendor demonstrates it, and retain evidence from the POC. Do not interpret a roadmap promise as a delivered capability.

Vendor: _____ Evaluator: _____ Date: _____

Scoring guidance: **0 = unavailable** | **1 = roadmap/custom** | **2 = partially supported** | **3 = demonstrated in scope**
A critical Required item scored below 3 should be documented as a gap or exception.

1. Asset Discovery, Inventory, and Coverage

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Automatically discover workloads across on-premises, colocation, AWS, Azure, GCP, and Kubernetes	☐	☐		
Inventory managed and unmanaged resources; identify blind spots before enforcement	☐	☐		
Capture cloud metadata, tags/labels, VPC/VNET, subnet, IP, hostname, namespace, pod, node, and application context	☐	☐		
Evaluate the ability to capture real-time workload metadata, including deep integrations with cloud provider services like IMDS and other native cloud services	☐	☐		
Assess whether the discovery and inventory data is presented in a unified console that supports both macrosegmentation and microsegmentation views, and facilitates the delivery of policy recommendations	☐	☐		
Measure discovery freshness and the time required for newly created workloads to appear	☐	☐		
Assess coverage for VMs, bare metal, containers/Kubernetes, ephemeral workloads, and serverless where applicable	☐	☐		
Evaluate CMDB/cloud inventory synchronization and reconciliation	☐	☐		

POC evidence/limitations:

_____

2. Traffic Flow Visibility and Application Dependency Mapping

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Observe workload-to-workload and process/ service communications in near real time	☐	☐		
Map clients, servers, ports, protocols, and application dependencies	☐	☐		
Visualize flows at host-to-host, group-to-group, application-tier, and cross-cloud levels	☐	☐		
Support historical flow analysis to account for infrequent jobs, batch processes, backup, patching, and failover paths	☐	☐		
Export or stream flow telemetry for SIEM, data lake, NDR, and troubleshooting workflows	☐	☐		

POC evidence/limitations:

---

3. Workload Identity, Context, and Dynamic Grouping

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Move beyond IP-only policy by using workload identity and contextual attributes	☐	☐		
Create dynamic groups from cloud tags, labels, application role, environment, ownership, location, namespace, or other metadata	☐	☐		
Automatically update membership as workloads scale, move, or are redeployed	☐	☐		
Evaluate resilience to poor or inconsistent tagging and the ability to combine multiple attributes	☐	☐		
Determine whether common policy objects can be reused across segmentation and adjacent security controls	☐	☐		

POC evidence/limitations:

---

4. Policy Modeling and Least-Privilege Design

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Support intuitive source-to-destination/service policy models without requiring deep network redesign	☐	☐		
Model allowlists for required application dependencies and default-deny for unnecessary communication	☐	☐		
Support host-to-host, group-to-group, and app-tier policy	☐	☐		
Provide policy templates or reusable service/application objects	☐	☐		
Evaluate rule ordering, precedence, conflict detection, shadowed rules, exceptions, and policy inheritance	☐	☐		
Assess whether policy can follow the workload rather than the subnet	☐	☐		

POC evidence/limitations:

---

5. AI/ML-Assisted Policy Recommendations

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Analyze observed flows to recommend workload groups and candidate least-privilege rules	☐	☐		
Identify uncovered assets or communications that fall outside intended segmentation policy	☐	☐		
Explain why a recommendation exists and which flows support it	☐	☐		
Provide confidence, impact, and exception context rather than opaque auto-enforcement	☐	☐		
Require human review/approval and auditable change history for AI-generated policy	☐	☐		

POC evidence/limitations:

---

6. Enforcement Architecture and Granularity

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Understand exactly where enforcement occurs: host, kernel, hypervisor, network, proxy/ gateway, cloud control, or combinations	☐	☐		
Validate that these converged security controls can be applied without requiring separate agents or complex traffic routing, distinguishing a unified platform from a collection of point products	☐	☐		
Evaluate OS-native enforcement mechanisms and their behavior under agent/service failure	☐	☐		
Support inbound, outbound, and internal controls where required	☐	☐		
Assess supported enforcement for TCP/UDP, IPv4/IPv6, ports, and service definitions; distinguish visibility metadata from actual policy enforcement capabilities	☐	☐		
Validate local enforcement without unnecessary traffic hairpinning or cloud egress	☐	☐		

POC evidence/limitations:

---

7. Kubernetes and Cloud-Native Segmentation

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Discover clusters, nodes, pods, namespaces, labels, services, and relevant runtime context	☐	☐		
Apply policy at a level that survives pod churn and autoscaling	☐	☐		
Distinguish visibility-only capabilities from enforceable Kubernetes segmentation	☐	☐		
Test east-west controls within a cluster, across clusters, and between Kubernetes and VM/database tiers	☐	☐		
Assess eBPF or other kernel telemetry/ enforcement use and operational implications	☐	☐		

POC evidence/limitations:

---

8. Hybrid and Multicloud Consistency

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Use a common policy model across data center, colocation, AWS, Azure, GCP, and cloud regions	☐	☐		
Protect cloud-to-cloud, cloud-to-data center, region-to-region, and VPC/VNET-to-VPC/VNET communication	☐	☐		
Avoid separate policy languages and management planes for each cloud wherever possible	☐	☐		
Evaluate how cloud metadata differences are normalized into a common policy abstraction	☐	☐		
Assess support for migrations where application components temporarily span environments	☐	☐		

POC evidence/limitations:

---

9. Zero Trust Platform and Adjacent Control Integration

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Evaluate integration between workload microsegmentation and user-to-application zero trust access	☐	☐		
Determine whether workload identity/context can be reused by SWG, firewall, DLP, TLS inspection, ZTNA, or other controls	☐	☐		
Assess interoperability with existing NGFW, cloud firewall, EDR/CWPP, CNAPP, IAM, PAM, and service-mesh investments	☐	☐		
Avoid duplicated enforcement and inspection that add latency or operational ambiguity	☐	☐		
Define which control plane owns macrosegmentation vs. microsegmentation	☐	☐		

POC evidence/limitations:

---

10. Policy Simulation, Staging, and Safe Rollout

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Provide observe/visibility-only modes before blocking	☐	☐		
Simulate proposed deny rules against historical and current flows	☐	☐		
Show which applications, hosts, processes, or business services would be affected before enforcement	☐	☐		
Support staged rollout by application, environment, resource group, or business unit	☐	☐		
Provide rapid rollback, emergency bypass, and time-bound exception mechanisms	☐	☐		
Test failover, DR, maintenance, backup, and rare application workflows before default-deny	☐	☐		

POC evidence/limitations:

---

11. Change Management, Automation, and Policy as Code

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Expose APIs and automation interfaces for inventory, policy, reporting, and lifecycle operations	☐	☐		
Assess available APIs and supported integrations with infrastructure automation and deployment workflows; distinguish shipping integrations from roadmap commitments	☐	☐		
Support versioning, approvals, peer review, rollback, and separation of duties	☐	☐		
Evaluate how changes to workload metadata or group membership affect applied segmentation policy and how administrators can investigate discrepancies	☐	☐		
Assess whether policy changes can be tested and promoted consistently from development to production	☐	☐		

POC evidence/limitations:

---

12. Agent Life Cycle, Resilience, and Tamper Protection

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Evaluate agent footprint, CPU/memory overhead, upgrade model, and compatibility with supported operating systems	☐	☐		
Support centralized rollout through enterprise software/configuration management	☐	☐		
Document supported enforcement behavior during agent disconnection or control-plane outages, and test observable effects on existing and new connections	☐	☐		
Protect enforcement components from unauthorized stop/kill/tampering	☐	☐		
Support staged agent upgrades (pilot group before broad deployment), maintenance scheduling, clear error reporting, and visibility into installed agent versions	☐	☐		
Document kernel/OS dependencies and interactions with EDR, host firewall, and hardening controls	☐	☐		

POC evidence/limitations:

---

13. Performance, Scale, and Availability

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Establish tested limits for workloads, agents, groups, policies, flows, events, and concurrent changes	☐	☐		
Measure policy propagation and convergence time during autoscaling and cloud churn	☐	☐		
Quantify latency and throughput impact for enforcement paths	☐	☐		
Validate availability and enforcement behavior during controller/service outages	☐	☐		
Assess regional/global architecture, data residency, and management-plane dependencies	☐	☐		

POC evidence/limitations:

---

14. Monitoring, Troubleshooting, and Security Operations

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Provide searchable flow logs, policy decisions, blocked/allowed events, and effective-policy views	☐	☐		
Answer "Why was this flow blocked/allowed?" quickly enough for operations teams	☐	☐		
Correlate workload metadata, policy, process/service, and traffic evidence in one investigation workflow	☐	☐		
Integrate alerts and telemetry with SIEM/ SOAR and incident-response processes	☐	☐		
Support packet/flow evidence or equivalent troubleshooting data where appropriate	☐	☐		
Measure mean time to diagnose segmentation-related application failures	☐	☐		

POC evidence/limitations:

---

15. Governance, RBAC, Audit, and Compliance

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Provide granular administrative RBAC and separation of policy authoring, approval, and operations	☐	☐		
Maintain immutable/auditable records of policy and administrative changes	☐	☐		
Define export options for logs and audit evidence	☐	☐		

POC evidence/limitations:

---

16. Operational Simplicity, Licensing, and TCO

Evaluation Requirement	Required	Desired	Score 0–3	Evidence/Notes
Compare deployment effort, infrastructure requirements, management planes, and staffing overhead	☐	☐		
Understand pricing units: workload, host, core, flow/traffic, cloud consumption, feature tier, or flat subscription	☐	☐		
Identify separately licensed visibility, recommendation, enforcement, Kubernetes, reporting, or integration capabilities	☐	☐		
Model three-year costs including agents, gateways/appliances, cloud egress, logging, professional services, and operations	☐	☐		
Assess whether consolidation genuinely retires point products or simply adds another control layer	☐	☐		
Evaluate time-to-value: discovery, first policy, first protected application, and expansion to enterprise scale	☐	☐		

POC evidence/limitations:

POC Exit Criteria

- [ ] **Baseline:** Observe representative traffic over a window that includes scheduled jobs, backups, maintenance, and known failover paths.
- [ ] **Discover and group:** Inventory workloads, validate owners, and test how IP, tag, and instance changes affect dynamic membership.
- [ ] **Recommend:** Generate candidate rules from observed traffic and require a knowledgeable administrator to validate AI-assisted suggestions.
- [ ] **Simulate:** Predict the impact of a proposed deny rule, document exceptions, and secure application-owner approval.
- [ ] **Enforce:** Apply restricted policy to the test application and confirm that authorized dependencies continue to work.
- [ ] **Attack-path test:** Attempt unauthorized connections between peers, tiers, environments, and administrative services.
- [ ] **Failure test:** Interrupt agent, controller, or cloud connectivity and record behavior for existing and new connections.
- [ ] **Operate:** Investigate an intentional false block, roll back safely, export evidence, and review exception expiration.
- [ ] **Measure:** Record performance overhead, policy propagation, analyst effort, and application-owner participation.

Decision Summary

Critical required gaps and compensating controls:

Supported platforms and deployment constraints:

License and three-year TCO assumptions:

Operational owners and unresolved dependencies:

Final recommendation and conditions for approval:

About the Author



Dave Shackleford

SANS Senior Instructor

Founder at Voodoo Security

Dave Shackleford has spent more than two decades working in cybersecurity across consulting, enterprise architecture, and security leadership. As CEO of Voodoo Security and a Senior Instructor with SANS Institute, he brings practitioner experience as a teacher and author for the SANS Sponsorship program. His work in cloud security, virtualization security, penetration testing, and enterprise defense gives students exposure to defensive techniques and real-world security operations.

COURSES TAUGHT

SEC504:
Hacker Tools, Techniques, and Incident Handling

SEC501:
Applied Cyber Defense

READ MORE ABOUT DAVE

Thank You to Our Sponsor

zscaler™

Thank you to Zscaler for supporting this SANS Buyer's Guide and helping make this research available to the cybersecurity community.

Learn more about Zscaler and microsegmentation:

- Get a full review of Zscaler Microsegmentation for workload security
- Learn how to segment your hybrid cloud with AI-powered microsegmentation
- See Zscaler Microsegmentation in action

About the SANS Research Program

The SANS Research Program is a key initiative by the SANS Institute and a premier global provider of cybersecurity research and information. SANS Research Program is designed to provide cybersecurity practitioners and leaders with data-driven insights, thought leadership, and solutions that help them better understand and respond to evolving security challenges.

All content is authored by SANS instructor experts from around the world who apply their years of experience from hands-on practitioner work in the field, advisory roles, and the classroom to provide education, guidance, and actionable insights that help make the cyber world a safer place.

To learn about sponsorship opportunities for research, content, and in-person or virtual events, email us at **Sponsorships@sans.org** or go to **www.sans.org/sponsorship**.

SANS | Research Program

© 2026 SANS™ Institute. All Rights Reserved.