100 Controls For Agentic AI Security

Governance and Risk Management

1. Executive accountability for Agentic AI
Implementation: assign an executive owner; publish a charter; tie KPIs to risk reduction.
Mitigates: unclear ownership, drift.

2. Agentic AI risk taxonomy
Implementation: define risk types specific to agents (prompt injection, rogue tools, model theft, emergent behavior); maintain examples.
Mitigates: blind spots, inconsistent triage.

3. System-level threat modeling for agents
Implementation: STRIDE+ATLAS profiling for planners, tools, memory, connectors; update each release.
Mitigates: unaddressed attack paths.

4. Risk acceptance workflow
Implementation: formal sign-off for residual Agentic AI risks with business justification and time-boxed exceptions.
Mitigates: shadow go-lives, unmanaged exposure.

5. Policy set for Agentic AI
Implementation: publish policies for autonomy limits, tool access, data usage, model use, and human-in-the-loop.
Mitigates: inconsistent behaviors, misuse.

6. AI use case intake and approval
Implementation: gated intake with risk scoring, PII sensitivity, tool reach, external calls, and blast radius.
Mitigates: risky deployments, scope creep.

7. Security requirements traceability
Implementation: baseline control requirements per use case; track to tests and evidences.
Mitigates: control gaps.

8. Third-party and supplier assurance for AI
Implementation: AI-specific TPRM questions for models, vector DBs, tool APIs, brokers; require pen test and SBOM.
Mitigates: supply chain compromise.

9. Safety-security trade-off review
Implementation: joint review of guardrails, content filters, and mission goals; document decisions.
Mitigates: over-permissive prompts, unsafe outputs.

10. Autonomy tiering and approval
Implementation: classify agent autonomy from 0 to 3; require stronger controls for higher tiers.
Mitigates: runaway actions, unsafe self-delegation.

11. Secure development lifecycle for Agentic AI
Implementation: add AI checkpoints into SDLC gates for data, prompts, tools, evals, and red teaming.
Mitigates: late discovery of risks.

12. KPI and KRIs for AI risk
Implementation: define leading indicators like prompt-inj attempts blocked, unsafe tool calls averted, model exfil attempts.
Mitigates: unknown risk posture.

Data Security and Privacy

13. Data classification for AI flows
Implementation: label inputs, memory, logs, embeddings; block restricted classes from model context.
Mitigates: accidental disclosure.

14. Context minimization
Implementation: enforce least data in prompts and memory; token-budgeted masking.
Mitigates: leakage via responses or tool calls

15. PII and secret scrubbing
Implementation: pre-prompt scrubbing, masking, or tokenization; test with seeded data.
Mitigates: data exposure, model memorization.

16. Training and fine-tune hygiene
Implementation: curated datasets, poisoning detection, lineage tracking, and quarantine of untrusted data.t
Mitigates: data poisoning, bias drift.

17. Embedding store protections
Implementation: encrypt at rest, row-level ACLs by tenant and purpose, query-time filters.
Mitigates: cross-tenant data bleed.

18. No-train and retention controls
Implementation: set model flags or proxies to disable training on production conversations; enforce retention windows.
Mitigates: unintended model learning.

19. Confidential compute for sensitive inference
Implementation: TEEs or VPC-scoped inference for regulated workloads.
Mitigates: cloud insider and co-tenancy risk.

20. Differential privacy for analytics
Implementation: DP libraries for reporting on agent runs; audit epsilon budgets.
Mitigates: re-identification.

21. Watermarking and content authenticity
Implementation: attach provenance metadata to AI outputs; verify inbound content signatures where supported.
Mitigates: spoofing, deepfake input.

22. Dataset access reviews
Implementation: quarterly access review for all data sources agents can query.
Mitigates: privilege creep.

Model and Prompt Security

23. Prompt registry with version control
Implementation: central store, approvals, diffs, rollback; map prompts to use cases.
Mitigates: prompt drift, hidden backdoors.

24. Prompt hardening patterns
Implementation: instruction isolation, deny-lists, meta-prompt self-checks, tool preconditions.
Mitigates: prompt injection, jailbreaks.

25. Anti-prompt injection filters
Implementation: pre- and post-processing to neutralize injection tokens, system override attempts, and tool-call coercion.
Mitigates: instruction override.

26. Output handling guardrails
Implementation: schema validation, type checks, regex safelists, and allow-only function names.
Mitigates: insecure output handling.

27. Model selection policy
Implementation: choose models by trust level, privacy guarantees, eval results; disallow unknown origins.
Mitigates: model risk, licensing traps.

28. Sensitive capability gating
Implementation: require explicit flags for code execution, file write, or network calls; add human approval for Tier 3 autonomy.
Mitigates: unsafe actions.

29. Model redaction responses
Implementation: forced refusals for sensitive queries; templated safe-completion fallbacks.
Mitigates: data leakage, misuse.

30. Model access isolation
Implementation: per-use case API keys, separate projects, and VPCs; no shared tokens.
Mitigates: blast radius.

31. Eval suite for security
Implementation: continuous tests for injection, data exfil, tool abuse, hallucinated APIs; publish scores.
Mitigates: regression risk.

32. Temperature and tool-use limits
Implementation: cap randomness and chain depth when tools are enabled
Mitigates: erratic behavior, looping.

Tooling and Function Use Security

33. Tool registry and attestation
Implementation: inventory every tool with purpose, inputs, side effects, auth model, and risk rating.
Mitigates: unknown capabilities.

34. Tool permission contracts
Implementation: granular scopes per tool function; enforce preconditions and max impact.
Mitigates: over-privilege.

35. Dry-run and simulation mode
Implementation: simulate tool calls with harmless stubs in test and lower envs; compare to expected.
Mitigates: unsafe execution.

36. Human-in-the-loop approval for destructive actions
Implementation: explicit confirmation UI with diff, impact estimate, and rollback plan.
Mitigates: irreversible changes.

37. Network egress controls for tools
Implementation: proxy allow-lists, DLP on egress, per-tool routing
Mitigates: data exfiltration, C2.

38. Rate limiting and cost guards
Implementation: per-tool quotas, concurrency caps, cost ceilings, circuit breakers.
Mitigates: model DoS, bill shock.

39. Command and code execution sandboxes
Implementation: ephemeral containers, syscall filters, filesystem jails, outbound blocks by default.
Mitigates: RCE fallout.

40. Tool input validation and canonicalization
Implementation: strict schema, path normalization, escaped shell args.
Mitigates: injection to downstream systems.

41. Tool output taint tracking
Implementation: mark untrusted outputs; require sanitization before reuse in prompts or other tools.
Mitigates: chained injection.

42. Tool health and integrity probes
Implementation: heartbeat checks, checksum of tool binaries or endpoints, version pinning.
Mitigates: supply chain tampering.

Agent Orchestration and Autonomy

43. Planner constraints
Implementation: cap steps, recursion, and parallel branches; enforce goal and scope limits.
Mitigates: runaway plans, lateral risk.

44. Role-based multi-agent segregation
Implementation: separate planner, critic, executor identities with scoped capabilities.
Mitigates: single-agent compromise impact.

45. Delegation policy
Implementation: explicit rules for spawning sub-agents, task scope, and hand-off approvals.
Mitigates: privilege escalation via delegation.

46. Memory governance
Implementation: TTLs for episodic memory, quality scoring, and removal of toxic or sensitive content.
Mitigates: long-term leakage, bias lock-in.

47. Goal alignment checks
Implementation: pre- and mid-run checks that steps map to stated objectives; abort on deviation.
Mitigates: off-policy actions.

48. Reward shaping safeguards
Implementation: penalize risky tool use, unsourced claims, and privacy violations in feedback loops.
Mitigates: misaligned optimization.

49. Constraint reminder meta-prompts
Implementation: inject constraints at each step, not only at start.
Mitigates: erosion of instructions.

50. Safety critic agent
Implementation: independent reviewer agent with veto on sensitive outputs and actions.
Mitigates: unsafe completions, tool misuse.

51. Explainability snapshots
Implementation: capture plan graphs and decision rationales for high-risk runs.
Mitigates: audit gaps, incident forensics friction.

52. Kill-switch and safe stop
Implementation: immediate halt command, revoke tokens, cancel queued tool calls.
Mitigates: damage containment.

Identity, Secrets, and Access

53. Agent identity distinct from users
Implementation: separate service principals for agents; no reuse of human tokens.
Mitigates: attribution gaps.

54. Ephemeral, scoped credentials
Implementation: short-lived tokens bound to a single run and tool; automatic rotation.
Mitigates: token replay, sprawl.

55. Delegation chain tracking
Implementation: propagate caller identity and purpose across sub-agents and tools.
Mitigates: non-repudiation gaps.

56. Policy-as-code for agent permissions
Implementation: OPA or ABAC policies referencing run context, data class, and autonomy tier.
Mitigates: static RBAC over-grant.

57. Secrets isolation and just-in-time access
Implementation: brokered secrets issuance per step; no secrets in prompts or memory.
Mitigates: leakage and theft.

58. Device and environment attestation
Implementation: require verified runtime for agents that execute code or handle sensitive data.
Mitigates: tampering, side-loading.

59. MFA proxies for human approvals
Implementation: step-up auth for destructive approvals; signed attestations.
Mitigates: social engineering.

60. Outbound identity for external APIs
Implementation: dedicated API identities and IPs; monitor for anomalous use.
Mitigates: cross-system impersonation.

Monitoring, Telemetry, and Detection

61. Agent run logging
Implementation: structured logs for prompts, tool calls, inputs, outputs, decisions, and approvals with hashes.
Mitigates: forensic gaps.

62. Security analytics for AI signals
Implementation: SIEM use cases for injection patterns, unusual tool sequences, exfil indicators.
Mitigates: stealthy attacks.

63. Model interaction anomaly detection
Implementation: baseline per use case; flag spike in refusals, token usage, recursion depth.
Mitigates: active exploitation.

64. Data egress DLP for agents
Implementation: pattern and policy checks post-generation and pre-egress.
Mitigates: data leakage.

65. Cost and quota monitoring
Implementation: alerts for cost anomalies, token floods, or atypical API regions.
Mitigates: DoS, abuse, fraud.

66. Content risk scoring
Implementation: score outputs for toxicity, PII, legal, and compliance; gate delivery.
Mitigates: harmful or illegal outputs.

67. Tool integrity monitoring
Implementation: verify tool endpoints, binaries, and configs at runtime.
Mitigates: supply chain tampering.

68. Provenance capture
Implementation: store source links, evidence, and reasoning for critical outputs.
Mitigates: unverified decisions.

Testing, Red Teaming, and Assurance

69. Security test plan per use case
Implementation: list of attack simulations, coverage matrix for prompts, tools, and data paths.
Mitigates: unknown weaknesses.

70. Adversarial red teaming
Implementation: scheduled ATLAS-informed campaigns targeting injection, tool abuse, exfil, and model theft.
Mitigates: real-world attack readiness.

71. Guardrail and filter testing
Implementation: measure bypass rates; maintain blocklists and allowlists with metrics.
Mitigates: brittle defenses.

72. Safety-security combined evals
Implementation: run safety and security evals together to catch trade-offs.
Mitigates: safety holes created by security changes and vice versa.

73. Pen testing of tools and brokers
Implementation: test function routers, sandboxes, and connectors like any other app surface.
Mitigates: classic web and API flaws.

74. Dataset health checks
Implementation: detect duplicates, outliers, poisoning artifacts; lock clean snapshots.
Mitigates: training-time compromise.

75. Pre-prod chaos drills
Implementation: inject failures in model calls, tool timeouts, and partial data.
Mitigates: resilience gaps

Incident Response and Resilience

76. AI-specific IR playbooks
Implementation: playbooks for injection, model key theft, tool compromise, data leakage; include roles and SLAs.
Mitigates: slow response.

77. Key rotation and model rebind
Implementation: rapid rotation for model and tool creds; update config management automatically.
Mitigates: token replay after breach.

78. Prompt and memory rollback
Implementation: versioned rollback of prompts, guardrails, and memory stores.
Mitigates: persistent compromise.

79. Containment levers
Implementation: throttle autonomy level, disable tool families, route to safe fallback models.
Mitigates: active damage.

80. Customer and regulator comms pack
Implementation: pre-approved templates, artifact list, and evidence bundle for AI incidents.
Mitigates: compliance and reputational impact.

81. Post-incident eval and guardrail tuning
Implementation: feed lessons into prompts, tools, policies, and tests; close the loop.
Mitigates: recurrence.

82. Resilience testing of kill-switch
Implementation: periodic drills to ensure safe-stop works under load.
Mitigates: fail-open risk.

Supply Chain, Deployment, and Infrastructure

83. SBOM and MBOM for AI stack
Implementation: IaC and Git-Ops for prompts, tool policies, and guardrail configs.
Mitigates: drift, undocumented changes.

84. Model artifact integrity
Implementation: signed checkpoints, hashes, and controlled registries.
Mitigates: model tampering.

85. Secure inference perimeter
Implementation: signed checkpoints, hashes, and controlled registries.
Mitigates: model tampering.

86. Config and prompt as code
Implementation: IaC and Git-Ops for prompts, tool policies, and guardrail configs.
Mitigates: drift, undocumented changes.

87. Environment parity
Implementation: reproduce prod runtimes locally with masked data and stubbed tools.
Mitigates: release surprises.

88. Cost and quota guardrails in infra
Implementation: infrastructure-level quotas, per-project budgets, and pre-commit checks.
Mitigates: economic DoS

89. TLS, mTLS, and cert pinning
Implementation: enforce mTLS between agent services, brokers, and tools; rotate certs.
Mitigates: MITM.

90. Geo and residency controls
Implementation: pin model regions and data stores; block cross-region spillover.
Mitigates: regulatory breach.

Human Factors, Ops, and Compliance

91. Human oversight training
Implementation: teach reviewers how to spot injection, hallucinated tools, and unsafe plans.
Mitigates: rubber-stamping risky actions.

92. Separation of duties for approvals
Implementation: different humans approve high-risk steps than those who authored prompts or tools.
Mitigates: insider risk.

93. Change management for prompts and tools
Implementation: ticketing, peer review, and automated tests before merge
Mitigates: accidental regressions.

94. Usage transparency for end users
Implementation: disclose agent capabilities, data use, and escalation paths.
Mitigates: long-term leakage, bias lock-in.

95. Legal guardrails for output use
Implementation: licensing checks, citation requirements, and high-risk domain disclaimers
Mitigates: IP and regulatory violations.

96. Accessibility and bias reviews
Implementation: test outputs for bias, fairness, and accessibility for key user groups.
Mitigates: ethical and legal risk.

97. Audit readiness pack
Implementation: keep artifacts for prompts, evals, red team results, incidents, and approvals.
Mitigates: failed audits, inability to prove control.

98. Decommissioning and model retirement
Implementation: revoke access, wipe memory stores, archive artifacts, and update docs.
Mitigates: orphaned risk.

99. User feedback loop and abuse reporting
Implementation: in-product reporting, triage, and hotfix pipelines for unsafe outputs.
Mitigates: prolonged exposure.

100. Continuous improvement council
Implementation: monthly cross-functional review of metrics, incidents, red team learnings, and roadmap.
Mitigates: stagnation and drift.

Copyright © Ampcus Cyber   www.ampcuscyber.com