



Architecting Secure Enterprise AI Agents with Salesforce and Model Context Protocol (MCP) A Reference Architecture and Risk-Governance Framework for Enterprise Agentic CRM

Rajesh Ediga

Wipro Ltd, Manuscript prepared for journal submission | September 2026

Abstract

Enterprise adoption of generative AI is moving from conversational assistants toward autonomous agents capable of reading enterprise data, invoking business logic, and performing operational actions. In a Salesforce environment, this evolution creates a security problem that differs materially from conventional API integration: an AI agent can interpret natural-language intent, select tools dynamically, construct parameters, and execute multi-step workflows, while the underlying CRM contains highly sensitive customer, financial, and operational information. Model Context Protocol (MCP) provides a standardized interaction layer between AI applications and external capabilities, but protocol-level connectivity does not by itself solve identity, authorization, data minimization, action governance, or audit requirements. This paper proposes a security-oriented reference architecture for enterprise AI agents interacting with Salesforce through MCP. The architecture combines per-user authentication, least-privilege authorization, Salesforce-native object and field controls, tool-level governance, context minimization, human approval for high-impact actions, continuous audit, and a five-factor risk model covering data impact, security impact, automation impact, integration impact, and reversibility. The paper maps these controls to current Salesforce Hosted MCP security capabilities and contemporary AI security guidance, then defines a practical evaluation framework for testing confidentiality, integrity, authorization correctness, prompt-injection resilience, operational safety, and auditability. The contribution is a reusable architectural pattern for enterprises that want to expose Salesforce capabilities to agentic AI without replacing established CRM governance with an opaque AI-specific security layer.

Keywords: Salesforce, Model Context Protocol, MCP, enterprise AI, agentic AI, AI security, authorization, OAuth 2.0, least privilege, prompt injection, CRM, governance, auditability, risk management

1. Introduction

Enterprise CRM platforms are increasingly becoming execution environments for AI-assisted work. Traditional business applications generally separate the user interface from the system of record through deterministic screens, APIs, validation rules, workflow engines, and access controls. Agentic AI changes this interaction pattern. Instead of a user selecting a menu, completing a form, and submitting a transaction, an AI agent can interpret a natural-language objective, discover available tools, retrieve context, reason over the retrieved information, and invoke one or more actions. This creates a new control problem: the enterprise must secure not only an API call, but also the decision path that led to the call.

Salesforce is a useful environment for studying this problem because its security model is already multi-layered. Identity is established through authentication; access is constrained by object permissions, field-level security, record sharing, profiles, and permission sets; and business behavior may be implemented through Flow, Apex, validation rules, approvals, and integrations. Salesforce Hosted MCP Servers extend this model to AI clients through the open Model Context Protocol. Salesforce documentation states that Hosted MCP transactions execute as the authenticated user and are constrained by the user's permissions, including object access, field-level security, and sharing rules [2], [4]. The current Salesforce approach also uses External Client Apps (ECAs) for MCP authentication rather than new Connected Apps [3].

MCP itself is designed as an open protocol for connecting AI applications to tools, resources, and prompts. The protocol has evolved rapidly; the July 28, 2026 specification introduced a more stateless protocol core and additional authorization hardening [1]. These capabilities improve interoperability, but interoperability also expands the attack surface. A malicious instruction in retrieved data, an over-permissioned tool, a stolen token, unsafe tool selection, or an ambiguous business command can convert a seemingly harmless conversational request into an enterprise-impacting action.

This paper therefore treats secure Salesforce-MCP integration as an architectural governance problem rather than simply a connectivity problem. The proposed architecture preserves Salesforce as the authoritative enforcement boundary while adding agent-specific controls around tool discovery, context handling, action approval, risk scoring, and audit. The objective is not to make an AI agent inherently

trustworthy; it is to design a system in which the agent remains bounded by deterministic enterprise controls even when model behavior is uncertain.

The paper makes four practical contributions:

- 1) a reference architecture that connects an AI agent to Salesforce through MCP while retaining Salesforce-native authorization;
- 2) a layered threat model for identity, tool use, context, data, actions, and integrations;
- 3) a five-factor action-risk model that can determine whether an agent should read, execute, request approval, or block an operation; and
- 4) an evaluation framework that can be implemented in a sandbox or controlled production pilot without assuming that successful connectivity implies secure autonomy.

2. Background And Related Work

2.1 Model Context Protocol

MCP standardizes how an AI application can discover and interact with external capabilities. At a high level, an MCP client connects a model-driven application to an MCP server that exposes tools and other contextual resources. The architectural value is interoperability: an AI client does not need a bespoke integration contract for every enterprise application. The 2026 specification also emphasizes improved authorization behavior and an architecture that can operate efficiently across ordinary HTTP infrastructure [1].

For enterprise security, however, protocol standardization is only one layer. MCP can describe how a tool is called, but the enterprise still has to decide which users can access that tool, what data the tool may expose, which side effects are permitted, and which activities require approval. These are policy questions that must be enforced independently of the model.

2.2 Salesforce security model for MCP

Salesforce Hosted MCP Servers are managed endpoints through which compatible AI clients can interact with Salesforce data and automation. Salesforce documentation describes multiple standard servers, including read-only and mutation-scoped SOBJect servers, and notes that each tool call is subject to the authenticated user's Salesforce permissions [5]. This creates an important design principle: MCP should not become a parallel authorization system that bypasses CRM governance. Instead, the agent should inherit the user context and operate within the existing security boundary.

Salesforce's current MCP setup uses an External Client App to establish OAuth-based trust between the client and Salesforce. Salesforce recommends controls such as PKCE, constrained user access, shortened refresh-token validity, IP restrictions where practical, and dedicated ECAs for distinct clients [3], [6]. These measures indicate that secure MCP deployment is an enterprise identity-management problem as much as it is an AI problem.

2.3 AI security and governance

NIST AI RMF 1.0 organizes AI risk management around Govern, Map, Measure, and Manage and is intended to help organizations incorporate trustworthiness throughout the AI lifecycle [7]. OWASP's 2025 guidance for LLM applications highlights prompt injection, sensitive information disclosure, improper output handling, supply-chain risks, and related threats [8]. OWASP's emerging MCP-specific project further identifies threats such as token exposure, privilege escalation, tool poisoning, command injection, prompt injection through contextual payloads, insufficient authorization, lack of telemetry, shadow MCP servers, and over-sharing [9].

These sources suggest that secure agent architecture should not depend on a single control. Defense in depth is required because the attack surface crosses identity, protocol, model behavior, application logic, data, and operational monitoring.

3. Problem Statement And Security Objectives

The core problem can be expressed as follows: how can an enterprise allow an AI agent to perform useful Salesforce operations without allowing model uncertainty, excessive privileges, malicious context, or integration errors to bypass established business controls?

A secure design should satisfy five primary objectives:

Objective	Security question	Design target
Confidentiality	Can the agent retrieve information beyond the user's authorized scope?	Enforce user identity, OAuth scopes, object/FLS/sharing controls, and context minimization.
Integrity	Can the agent make an unintended or excessive change?	Constrain tools, validate parameters, use risk-based approval, and preserve transaction boundaries.
Availability	Can agent activity overload or disrupt Salesforce processes?	Use bounded toolsets, rate limits, query controls, and workload-aware orchestration.

Accountability	Can the organization determine who initiated and approved an action?	Preserve per-user identity and correlate prompts, tool calls, approvals, and Salesforce audit events.
Recoverability	Can harmful operations be reversed?	Prefer reversible actions, preconditions, staged changes, and explicit rollback plans.

4. PROPOSED REFERENCE ARCHITECTURE

The proposed architecture consists of eight logical layers. The design deliberately separates model reasoning from enterprise enforcement so that the language model is not treated as the final authority.

Figure 1. Secure Salesforce-MCP reference architecture

1. User / Business Request Natural-language objective, authenticated corporate identity
2. AI Application / Agent Runtime LLM, planner, memory, conversation state, tool-selection logic
3. Agent Security Gateway Policy checks, prompt/context filtering, tool allowlist, rate and cost controls
4. MCP Client Protocol client, OAuth flow, token handling, server discovery
5. Salesforce Hosted MCP Server Standard or custom tools exposed through governed MCP interfaces
6. Salesforce Security Boundary User permissions, permission sets, CRUD, FLS, sharing, validation, Flow, Apex
7. Enterprise Systems / Data CRM records, integrations, analytics, downstream applications

Security principle: the agent proposes; deterministic controls authorize; Salesforce remains the system of record and final permission boundary.

4.1 Layer 1: User identity

The user should be strongly authenticated before an agent is permitted to access enterprise CRM data. The architecture should avoid shared human identities because shared credentials weaken accountability and make least-privilege design difficult. For hosted Salesforce MCP, current guidance explicitly ties transactions to the authenticated user rather than a shared integration identity [6].

4.2 Layer 2: Agent runtime

The agent runtime performs reasoning and tool selection, but it should not have unrestricted authority. A useful design separates planning from execution. The model can construct a proposed action, while a policy component determines whether that action is permitted. This creates a deterministic security boundary around inherently probabilistic reasoning.

4.3 Layer 3: Agent security gateway

A security gateway is recommended even when Salesforce already enforces strong permissions. Its purpose is not to replace Salesforce authorization but to apply agent-specific controls: tool allowlists, context-size limits, action classification, sensitive-field filtering, approval rules, anomaly checks, request correlation, and rate controls. The gateway can also prevent the model from treating retrieved text as executable policy.

4.4 Layer 4: MCP client and OAuth

The MCP client is responsible for protocol communication and token lifecycle. Current Salesforce guidance requires an External Client App for Hosted MCP and supports OAuth-based authentication; PKCE and JWT-based mechanisms are described in the platform setup guidance [3], [6]. Tokens should be stored outside prompts, logs, model memory, and tool arguments. Token rotation and revocation should be part of the operational control plane.

4.5 Layer 5: MCP server

Tool exposure should follow the principle of least privilege. Salesforce provides server choices with differing capabilities, including read-only access and mutation-oriented access [5]. A read-only server is preferable for analytical use cases; mutation and deletion capabilities should be exposed only when the business process requires them. Tool descriptions should explain side effects, idempotency, expected parameter types, and whether the operation is reversible.

4.6 Layer 6: Salesforce security boundary

Salesforce should remain the authoritative authorization layer for CRM data. The architecture relies on existing controls such as CRUD permissions, field-level security, sharing rules, permission sets, validation rules, and business logic. Salesforce documents that Hosted MCP calls are evaluated against these controls [4], [5]. This dramatically reduces the need to create an independent copy of the data authorization model in the AI layer.

4.7 Layer 7: Downstream integrations

A CRM action can trigger flows, Apex, events, middleware, data synchronization, notifications, or external financial and operational systems. Therefore, an action that appears low-risk at the data level may be high-risk at the integration level. The architecture should detect or classify actions that can initiate downstream side effects and increase the required approval level accordingly.

5. THREAT MODEL

The threat model considers both malicious actors and benign-but-unsafe model behavior. The latter is important because an LLM can produce harmful actions without being explicitly compromised.

Threat	Example in Salesforce-MCP	Primary control
Prompt injection	A case description tells the agent to export unrelated customer records.	Treat retrieved content as untrusted data; apply tool and data policy outside the model.
Over-privileged tool	Agent receives full CRUD access when it only needs read access.	Use scoped MCP servers, permission sets, and tool allowlists.
Token exposure	Access token appears in logs, prompts, or telemetry.	Secret isolation, redaction, rotation, revocation, short validity.
Tool poisoning	A tool description or metadata attempts to influence model behavior.	Validate tool metadata, maintain approved tool registry, monitor changes.
Excessive action	Agent updates thousands of records due to ambiguous scope.	Volume thresholds, dry-run mode, approval, idempotency and transaction limits.
Context over-sharing	Agent retrieves sensitive fields irrelevant to the task.	Data minimization and field-level enforcement.
Integration cascade	One CRM update triggers unintended downstream processing.	Dependency mapping, risk scoring, approval and post-action monitoring.
Audit gap	Organization cannot determine who authorized an agent action.	Per-user identity, correlation IDs and event logging.

5.1 Prompt injection

Prompt injection is especially important because Salesforce records themselves may contain free-text fields such as case comments, opportunity notes, emails, descriptions, or knowledge content. Such content can contain instructions that look authoritative to an LLM but are not authorized policy. The architecture therefore treats all retrieved business content as data, not as instructions. Any permission decision must be made by deterministic control logic and Salesforce authorization, not by the text of a record.

5.2 Privilege escalation

Privilege escalation can occur when the agent is given a tool that is broader than the user task. A common anti-pattern is connecting an agent to a full CRM API because it is convenient during development. Production deployments should use the narrowest tool set that satisfies the business capability. Salesforce's current standard server catalog provides scoped choices such as SObject Reads and SObject Mutations, enabling this principle at the platform boundary [5].

6. IDENTITY, AUTHORIZATION, AND LEAST PRIVILEGE

6.1 Per-user authorization

The recommended authorization pattern is per-user access. The agent acts on behalf of a named user, Salesforce authenticates that user through the configured OAuth integration, and every operation is evaluated against that user's effective Salesforce permissions. Salesforce explicitly describes this model for Hosted MCP, including the application of CRUD, field-level security, and sharing [4], [6].

6.2 External Client App governance

An External Client App should be treated as a security boundary, not simply as configuration. Separate ECAs for distinct AI clients improve audit isolation and policy control. Access can be constrained to pre-authorized users through permission sets and application policies; current Salesforce guidance also describes IP restrictions and refresh-token controls as optional hardening measures [3].

6.3 Least privilege by capability

Least privilege should be applied at three levels: application access, MCP server capability, and Salesforce data permission. For example, a sales-insights agent may need read-only opportunity and account information but should not receive delete capability, setup administration, or unrestricted access to sensitive objects. Capability-specific servers and permission sets reduce the blast radius when the model makes a mistake.

7. CONTEXT SECURITY AND DATA MINIMIZATION

An enterprise agent does not need every field merely because the authenticated user can see it. The architecture should distinguish between what a user is permitted to access and what the agent needs to

complete a task. This is a data-minimization principle. Excessive context increases both confidentiality risk and the probability that irrelevant content will influence the model.

Context control	Purpose	Example
Field minimization	Reduce unnecessary exposure.	Exclude personal identifiers when only account segment is needed.
Record filtering	Keep retrieval scoped to the business request.	Only retrieve opportunities for a named region or account.
Semantic boundaries	Prevent cross-domain mixing.	Do not combine HR, legal, and sales records unless explicitly required.
Sensitivity labels	Apply stricter handling to protected fields.	Treat financial, identity, or regulated fields as restricted.
Output filtering	Prevent sensitive content from being echoed to users or logs.	Redact secret or restricted fields from generated summaries.

Context minimization also improves model reliability. Smaller, task-relevant context reduces ambiguity and makes the agent's reasoning easier to evaluate. However, minimization must not create a false security boundary: Salesforce permissions remain the authoritative enforcement point.

8. RISK-BASED ACTION GOVERNANCE

A central contribution of this paper is a practical risk model for deciding how much control an agent action requires. Each proposed Salesforce action is scored on five dimensions: Data Impact (D), Security Impact (S), Automation Impact (A), Integration Impact (I), and Reversibility (R). Each factor receives a value from 1 to 3, where 1 is low, 2 is medium, and 3 is high.

Risk Score = D + S + A + I + R

A baseline classification is: 5–7 = Low Risk, 8–11 = Medium Risk, and 12–15 = High Risk. Organizations can tune these thresholds after observing real workload patterns. The score should be supplemented by hard-stop rules for prohibited actions such as unauthorized access, unsafe credential handling, or destructive operations without required approval.

Factor	Low (1)	Medium (2)	High (3)
Data Impact	No meaningful data change.	Limited record changes.	Mass update/delete, schema change, or potential data loss.
Security Impact	No change to security.	Limited permission-related effect.	Changes access, sensitive data, or security policy.
Automation Impact	No automatic side effects.	Limited or user-initiated automation.	Triggers, flows, batches, approvals, or large automated cascades.
Integration Impact	No external impact.	Non-critical integration effect.	Critical API, middleware, financial, or operational integration.
Reversibility	Easy to undo.	Rollback requires controlled effort.	Hard to reverse or potentially irreversible.

8.1 Action classes

Action class	Typical score	Agent behavior
Informational	5–7	Execute automatically; log action.
Controlled mutation	8–11	Validate scope, preview changes, then execute under policy.
High-impact mutation	12–15	Require explicit user approval or workflow approval before execution.
Prohibited	Any score	Block regardless of score when the action violates policy or authorization.

Example: a request to summarize account activity is typically low risk. A request to update one opportunity may be medium risk. A batch operation that modifies hundreds of thousands of accounts and triggers downstream integrations can become high risk even if the user is otherwise authorized. The key insight is that risk is determined by the effect of the action, not simply by the fact that the action is an Apex call, Flow invocation, or MCP tool call.

9. HUMAN APPROVAL AND CONTROLLED AUTONOMY

Secure agent architecture should distinguish between autonomy and authority. An agent can be highly autonomous in planning while remaining constrained in execution. Human approval should be introduced when the potential consequence of an action exceeds the organization's acceptable risk threshold.

Condition	Recommended control
Read-only retrieval	Automatic execution when authorization is valid.
Small reversible update	Automatic execution with parameter validation and logging.
Large update	Preview affected records, require confirmation, enforce volume threshold.
Delete operation	Explicit approval and stronger preconditions; consider a scoped delete tool.
Security/configuration change	Require privileged approval or separate administrative workflow.
Cross-system financial/operational action	Require approval and post-action verification.

Approval should not rely on the agent asking the user in natural language and then interpreting a vague response as authorization. Instead, the approval should be represented as an explicit system event tied to a concrete action plan, scope, parameters, and timestamp. This creates an audit-ready chain from intent to authorization to execution.

10. AUDITABILITY AND OBSERVABILITY

Auditability is necessary for both security and operational learning. For each agent interaction, the system should be able to reconstruct who initiated the request, which model or agent version was used, which tools were considered, which tool was invoked, what scope and parameters were passed, whether approval was required, and what Salesforce transaction occurred.

Salesforce states that Hosted MCP activity is attributable to the authenticated user and is visible through its platform event logging and usage telemetry [4]. The recommended architecture adds correlation at the agent layer so that a conversation request can be linked to MCP calls and downstream Salesforce events. Sensitive prompt content should not be logged indiscriminately; logs should capture enough metadata for security investigation without creating a secondary data-leak surface.

Telemetry	Example fields	Retention principle
Request	Request ID, user, timestamp, agent version	Keep according to enterprise audit policy.
Policy decision	Risk score, rule, approval state	Retain for control validation and incident review.
MCP invocation	Server, tool, parameters hash, latency	Avoid raw secrets; retain useful operational metadata.
Salesforce outcome	Object, operation, result, error code	Link to CRM audit and event telemetry.
Approval	Approver, action fingerprint, timestamp	Retain as evidence of authorization.

11. SECURE DEVELOPMENT AND DEPLOYMENT LIFECYCLE

Security controls are strongest when applied before production deployment. A practical lifecycle is: design, build, test, pilot, monitor, and review.

Design

Document business capability, data classes, tool inventory, action boundaries, and risk thresholds.

Build

Implement least-privilege permissions, scoped MCP servers, secure secrets handling, deterministic policy checks, and correlation IDs.

Test

Use realistic positive and negative cases, including prompt injection, unauthorized access, excessive volume, malformed parameters, integration failures, and rollback tests.

Pilot

Start with read-only capabilities, a small user population, and explicit monitoring. Add mutation capabilities only after authorization behavior is validated.

Monitor

Review logs, policy violations, failed tool calls, unusual volumes, and changes in tool metadata.

Review

Periodically reassess permissions, token policies, tool inventories, model versions, and risk thresholds.

A useful operational rule is to treat MCP tool changes like API changes. A new tool, new tool description, newly exposed field, or expanded mutation scope can materially change the security posture and should move through change management, testing, and approval.

12. EVALUATION FRAMEWORK

This paper proposes an evaluation method rather than claiming empirical results. The architecture can be evaluated in a Salesforce sandbox or controlled pilot using a representative CRM dataset and a fixed set of agent tasks.

Dimension	Metric	Example test
Authorization correctness	Percent of unauthorized actions blocked	User without Account edit permission attempts update.
Data minimization	Percent of unnecessary sensitive fields exposed	Task requests account summary while restricted fields exist.
Prompt-injection resistance	Unsafe tool execution rate	Malicious instruction embedded in case text.
Action safety	High-impact actions executed without approval	Agent attempts large update or delete.
Audit completeness	Percent of actions traceable end-to-end	Trace request -> policy -> MCP -> Salesforce outcome.
Operational performance	Latency, tool failure rate, throughput	Repeat standardized agent workflows.
Recovery	Successful rollback rate	Inject controlled failure after multi-record update.

The evaluation should include a matrix of users, permission sets, data sensitivity levels, tool scopes, and action sizes. A useful test suite combines deterministic security tests with adversarial prompts. For example, the same business request can be executed under three user profiles: a read-only user, a limited business user, and an administrator. The expected result should follow Salesforce authorization consistently, regardless of the model's language or confidence.

12.1 Experimental hypothesis

The primary hypothesis is that a layered architecture—per-user identity + Salesforce authorization + tool scoping + context minimization + risk-based approval + audit correlation—will reduce the probability and blast radius of unsafe agent actions compared with an architecture in which an agent directly receives broad API access. A secondary hypothesis is that explicit tool scoping and context minimization will improve operational predictability by reducing irrelevant tool choices and unnecessary data retrieval.

12.2 Baseline comparison

A useful baseline is a conventional agent that receives a broad API capability with minimal external policy. The experimental architecture can then be compared against the baseline on unauthorized-action rate, sensitive-field exposure, excessive-volume actions, audit completeness, and task completion. Care should be taken not to interpret higher safety controls as automatically superior if they make legitimate tasks impossible; the objective is a measurable balance between utility and controlled autonomy.

13. REFERENCE USE CASES

Use case	Typical agent behavior	Security posture
Sales assistant	The agent retrieves open opportunities, summarizes pipeline risk, and drafts a follow-up plan.	Read-only MCP capabilities; no mutation without explicit request.
Service operations	The agent summarizes a case, checks entitlement-related data, and proposes a status update.	Permission-aware retrieval; mutation treated as controlled action.
Data maintenance	The agent identifies records with missing attributes and proposes updates.	Preview + batch limits + approval for bulk mutation.
Executive analytics	The agent produces a cross-object performance summary.	Strict context minimization and read-only access.

Administrative automation	The agent proposes a configuration or automation change.	Separate privileged workflow; human approval mandatory.
---------------------------	--	---

These scenarios illustrate a general design principle: not every agent needs the same level of autonomy. Enterprise architecture should classify capabilities according to business consequence and expose only the minimum capability set needed for each role.

14. Discussion

The proposed architecture intentionally keeps Salesforce at the center of authorization. This is significant because duplicating CRM security logic outside Salesforce creates synchronization risk: a user may be allowed to perform an action in one system but denied in another because policy copies have drifted. Per-user enforcement reduces this risk and allows existing Salesforce governance investments to remain relevant to AI agents.

At the same time, Salesforce authorization alone does not answer every agent-specific question. A user may legitimately have permission to update one record, yet an AI agent might attempt to update 100,000 records because of an ambiguous instruction. The five-factor risk model addresses this gap by assessing consequence, not only permission. This distinction is crucial: authorization answers “can this user do it?” while action governance asks “should this agent perform this specific operation under these circumstances?”

MCP also creates an ecosystem risk. Interoperability makes it easier to connect multiple clients, servers, tools, and agents, which can accelerate innovation but also complicate governance. Organizations should maintain an approved registry of MCP clients and servers, document ownership, manage configuration as code where appropriate, and monitor changes to tool definitions and scopes. OWASP’s MCP risk project highlights the relevance of token management, privilege escalation, tool poisoning, authorization gaps, telemetry gaps, and context over-sharing [9].

Finally, human oversight should be treated as an architectural mechanism rather than a conversational courtesy. When an action is high-impact, a structured approval event should be required. The model can explain its reasoning and proposed action, but the enforcement layer should decide whether approval is needed and ensure that the approved action matches what will actually execute.

15. LIMITATIONS AND FUTURE RESEARCH

This paper is a reference architecture and evaluation proposal, not a controlled empirical study. The risk thresholds are intentionally generic and should be calibrated using organization-specific incident history, compliance requirements, transaction volumes, and business criticality. The architecture also assumes that the underlying Salesforce security configuration is itself correct; a poorly governed CRM permission model can still expose sensitive data to an authorized agent.

Future research should measure agent behavior over larger workloads, compare different models and agent frameworks, evaluate tool-poisoning and contextual prompt-injection attacks, quantify the performance cost of policy enforcement, and study how risk scores can adapt from historical outcomes. Another promising direction is policy-as-code for agent actions, in which business rules are expressed independently of natural-language instructions and evaluated before every side-effecting tool call.

16. Conclusion

Secure enterprise AI agents require more than a standards-compliant connection to a CRM. Salesforce and MCP provide an important foundation for interoperability, identity, and enterprise authorization, but safe autonomy emerges from the combination of controls around the model and controls beneath it. The architecture proposed in this paper places the AI agent inside a layered security model: per-user authentication, least-privilege MCP capabilities, Salesforce-native authorization, context minimization, deterministic action policy, risk-based approval, and continuous audit.

The five-factor risk model—data impact, security impact, automation impact, integration impact, and reversibility—provides a practical mechanism for moving from binary “allowed or denied” decisions toward controlled autonomy. Low-risk tasks can remain fast and largely automatic, while high-impact operations can be routed to approval or blocked. This approach preserves the productivity benefits of agentic AI without making the language model the final security authority.

For enterprise architects, the central design lesson is simple: treat the agent as an untrusted planner operating inside a trusted control plane. Let the model propose; let deterministic policy evaluate; let Salesforce enforce; and let audit systems record what actually happened. That separation of reasoning and authority is the foundation for secure, scalable Salesforce-MCP deployments.

References

- [1] Model Context Protocol, “The 2026-07-28 Specification,” Model Context Protocol Blog, Jul. 28, 2026.
- [2] Salesforce Developers, “Hosted MCP Servers,” Salesforce Developer Documentation, 2026.

- [3] Salesforce Developers, "Create an External Client App," Hosted MCP Servers Developer Guide, 2026.
- [4] Salesforce Developers, "How to Secure Salesforce Hosted MCP Servers," Salesforce Developers Blog, Jun. 2026.
- [5] Salesforce Developers, "Standard MCP Servers Reference," Hosted MCP Servers Documentation, 2026.
- [6] Salesforce Developers, "Connecting an MCP Client," Hosted MCP Servers Documentation, 2026.
- [7] E. Tabassi, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, National Institute of Standards and Technology, Jan. 2023, doi: 10.6028/NIST.AI.100-1.
- [8] OWASP GenAI Security Project, "OWASP Top 10 for Large Language Model Applications 2025," 2025.
- [9] OWASP, "OWASP MCP Top 10," OWASP GenAI Security Project, 2025–2026.
- [10] Salesforce Developers, "General Best Practices," Hosted MCP Servers Documentation, 2026.