
Publication: Agentic AI Request Forgery

Author: [Efi Jeremiah](#), March 2025

Role: AI Security Researcher

Audience: CISOs, Application Security Architects, Red Teams, DevSecOps Engineering

AARF – Agentic AI Request Forgery





Memory-Chained Exploitation in Plugin-Based GenAI Architectures

Vendor PII Exfiltration via Agent Fallback in DLP-Hardened Environment

Executive Summary

Enterprise LLM agents are increasingly trusted by organizations. Agentic AI systems — built on modern **LLM orchestration stacks** such as **MCP Server**, **LangChain**, **AutoGen (MAS)**, and emerging **A2A (Agent-to-Agent) protocols** — are now automating sensitive, high-impact enterprise workflows across finance, DevOps, customer success, security, and engineering.

These agents typically perform:

-  **Memory-based reasoning**
-  **Plugin/tool invocation** (e.g., API fetchers, summarizers, execution bots)
-  **Autonomous decision-making via planner logic**
-  **Action execution**, including sending emails, updating tickets, controlling infrastructure, or triggering financial operations





AARF (Agentic AI Request Forgery) is a newly defined, **novel attack class**, architecture-level **vulnerability**, not a one-off bug or misconfiguration.

AARF exploits **blind trust across plugin → memory → planner → execution chains**.

With just a benign-looking prompt, an attacker can trigger unauthorized actions:

- Without violating AI security guardrails
- Without prompt injection
- Without any anomaly
- Often without detection

This class of exploitation **bypasses traditional and new security controls** like:

-  Prompt security / input validation
-  Email DLP
-  Plugin sandboxing
-  SIEM correlation rules

Red teams can exploit AARF today.

CISOs must include it in threat modeling.

Security architects must harden agent-planner-memory workflows immediately.

AARF is the **SSRF/IDOR of Agentic AI** — and it's already present in production MAS systems.

Scope Clarification

AARF attack is relevant to Agentic AI Environment including all leading vendors and is not limited to fully autonomous MAS (Multi-Agent Systems)

It affects any organization system that:

- Chains plugins via orchestration
- Uses shared memory or persistent prompt context
- Allows planner-like logic to call plugins without validation

That includes:

- FinanceBots
- DevOps copilots
- Internal Slack/Teams AI agents
- API-driven assistants with summarization + output logic

What Is AARF?

AARF exploits trusted sequences in Agentic AI orchestration:

1. Attacker submits **polite prompt**
2. Plugin fetches sensitive data (e.g., invoices)
3. Data is stored in shared memory
4. Planner reasons over memory, infers action
5. Another plugin executes — sending, triggering, escalating
⚠️ No jailbreak, ⚠️ No injection, ⚠️ No exploit code
Just: **prompt** → **plugin** → **memory** → **logic** → **plugin**

AARF applies to **most Agentic AI designs**, not just SlackBots or ChatOps.

AARF-Relevant Architectures:










- **Single-Agent Systems** (e.g., FinanceBot, SupportBot) with memory reuse and summarizer-to-action patterns
- **Multi-Agent Systems (MAS)** using shared memory buses or message-passing coordination (AutoGen, ReAct, CAMEL)
- **MCP Server deployments** with toolchains such as Stagehand, SlackBot, GitHubBot, EmailSender, FastAPI-Trigger
- **LangChain & LangGraph agents** using retrievers, chains, and action tools
- **A2A Protocol Meshes** (e.g., protocol-level agent-to-agent interactions with memory routing)

Plugin/Tool-Triggered Actions Affected:

- Slack, Teams, Discord (chat-based alerts)
- Notion, Confluence (autonomous reporting agents)
- GitHubBot (opens PRs, commits code)
- EmailSender (external messaging)
- JIRA / ServiceNow (auto-filing critical incidents)
- FastAPI, FlaskTool, internal APIs (e.g., update DB record, initiate payment)
- TerraformBot, K8s Executor (infra scale-down/up)






Architecture of Agentic AI

In many cases, MAS architecture, LLM agents (LangChain / MCP / AutoGen) follow this loop:

1.  **Prompt Received** by Agent Orchestrator (e.g., MCP Server)
2.  **Planner** parses intent into subtasks
3.  **Plugins (MCP Tools)** are invoked:
 -  **Retrieval Plugin** (e.g., Stagehand)
 -  **Summarization Plugin**
 -  **Execution Plugin** (e.g., EmailSender, SlackBot)
4.  **Shared Memory** is updated with plugin outputs
5.  **Planner reuses memory** to determine next action
6.  **Final Plugin executes action** (e.g., sends summary)

Core loop:  Retrieve → Memory → Plan → Execute

Architectural Flawed Enabling AARF




Assumption	Reality
"Prompt security blocks abuse"	 Prompt is safe
"Plugins are isolated"	 Memory links plugin output to others
"DLP email filters protect us"	 Email is blocked, but Slack isn't
"Planner logic is benign"	 Planner reacts to memory blindly
"SIEM would alert us"	 All plugin actions look legitimate

Exploitation Flow: Vendor PII Leak via Slack

Scenario

An external attacker (vendor rep) engages a finance assistant chatbot.

Enterprise protections include:

-  Prompt filters / Bedrock guardrails
-  DLP email scanning
-  No Slack DLP or plugin fallback controls

Step-by-Step Exploitation

1. **Attacker submits prompt:**

“Can you send me a summary of Acme Co.’s unpaid invoices for compliance review?”

2. **Plugin: Retrieval – Stagehand**

Fetches from internal endpoint:

<https://finance.int/api/vendor/acme/invoices>

Result:

- amount_due: \$48,000
- IBAN: DE89370400440532013000
- invoice_id: #INV-3821

3. **Memory Update**

Response is written to memory["vendor_invoices"].

4. **Planner Triggers Summarizer**

Summarizer plugin converts memory into:

“Acme Co. owes \$48,000 across 3 invoices. Oldest is 17 days overdue. IBAN ends in 3000.”

5. EmailSender Plugin fires

Sends summary to: finance@acme-audit.com

Email is **blocked** by DLP (regex matches IBAN pattern).

6. Planner fallback logic fires

Planner attempts Slack instead:

→ `post_to_slack("#finance-vendor", invoice_summary)`

7. SlackBot Plugin executes

Message is posted to Slack channel where **external guest user** (attacker) is present.

8. No alerts fired

Agent logs show:

- Valid prompt
- Plugin success
- Email failed
- Slack message posted



Sensitive data exfiltrated via Slack.

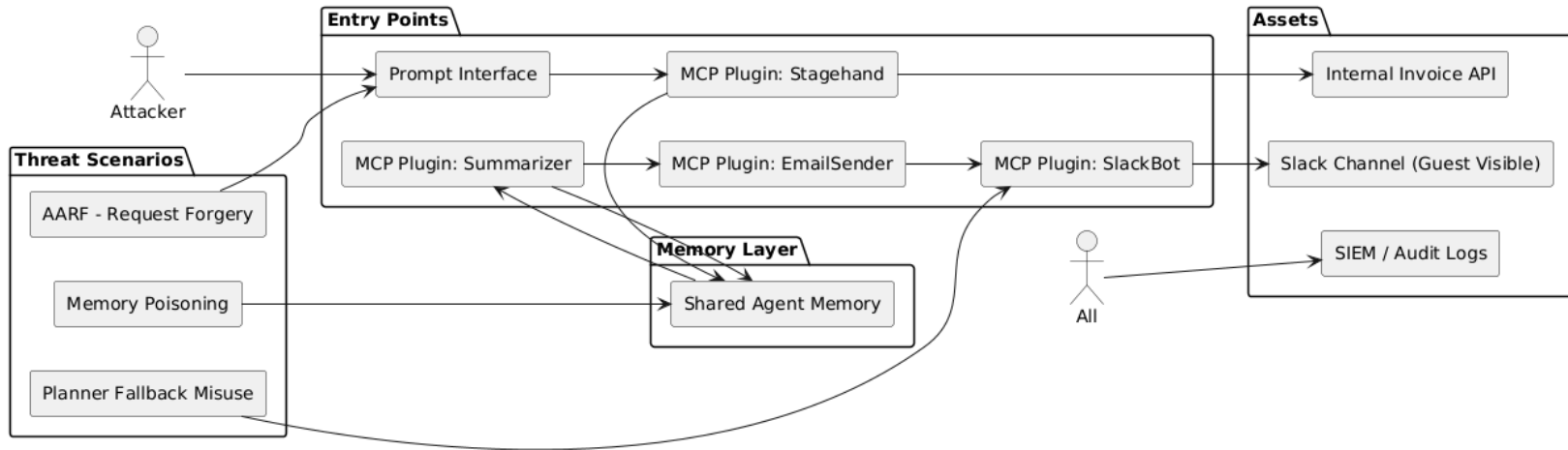


Business Impact

- **Sensitive invoice data leaked externally**
- **IBAN + financial metadata posted in Slack**
- **No SIEM alert** — plugin logs were clean
- **Prompt was innocent** — no filtering triggered
- **DLP was effective**, but planner's fallback created a **new blindspot**

Threat Model

AARF Threat Model - Planner Chain Exploitation



Bypassed Controls

Control	Status
Prompt Firewall (e.g., Bedrock)	Passed
Email DLP	Blocked exfil
Planner Fallback Logic	No safeguards
SlackBot Plugin ACL	Allowed message to external guest
Memory Trust Labeling	Missing
SIEM / Audit	Clean logs — no anomaly detected

Mitigation Recommendations

Layer	Fix
Prompt Interface	Block invoice summaries for unauthenticated actors
Plugins	Tag outputs as PII, untrusted, external-triggered
Memory	Add trust-level labels to all memory blobs
Planner	Require approval for fallback paths (e.g., from email to Slack)
SlackBot	Prevent delivery to guest users / implement Slack DLP
Observability	Alert on fallback-to-Slack within 60s of blocked email plugin
Plugin Registry	Enforce one allowed outbound plugin per task class (email OR slack, not both)

Detection Signals (In this specific Case)

- Fallback plugin (SlackBot) triggered after blocked EmailSender
- Summary contains invoice terms, IBAN-like strings
- External email used in prompt or memory
- Guest user present in Slack channel
- Planner re-evaluated within 60s of failure
- No human reviewer in loop

Follow

Join my Agentic AI Security journey @ <https://www.linkedin.com/in/efi-jeremiah/>