논문 2026-2-11 http://dx.doi.org/10.29056/jsf.2026.06.11

MCP 기반 모듈형 AI 업무 에이전트의 아키텍처 설계 및 구현

송재숙*, 배선영**†

Architecture Design and Implementation of MCP-Based Modular AI Work Agent

Jea-Suk Song*, Sun-Young Bae**†

요 약

최근 대규모 언어 모델(LLM:Large Language Model) 기반 AI 에이전트 생태계는 Model Context Protocol(MCP)의 표준화에 힘입어 자율형 업무 자동화 시스템으로 급격히 진화하고 있다. 그러나 기존 에이전트 프레임워크 연구들은 주로 고정된 단일·멀티에이전트 파이프라인 설계에 치중되어, 업무 환경의 요구사항에 따른 실시간 동적 모듈 조합 유연성이 부족하며, 과도한 도구 권한 부여에 따른 보안 제어력 상실 문제를 노정하고 있다. 본 연구는 이러한 학술적·기술적 한계를 극복하고 안전한 업무 자동화를 구현하기 위해, MCP를 표준 인터페이스로 적용하여 사용자가 필요한 기능만을 선택적으로 조합할 수 있는 모듈형 AI 업무 에이전트 아키텍처를 제안한다. 제안 아키텍처는 최소 코어(Minimal Core), 선언적 모듈 등록(Declarative Module Registry), 최소 권한 원칙(Least Privilege), Human-in-the-Loop의 4대 설계 원칙에 기반하며, 각 원칙은 ISO/IEC 25010 소프트웨어 품질 특성과 명시적으로 대응된다. 기존 도구와의 설계적 비교 분석, 선행 연구 보안 권고 매핑, 시나리오 기반 보행(Walkthrough) 분석을 통해 아키텍처의 유효성과 위협 차단 성능을 검증하였다. 본 연구는 에이전트 표준 프로토콜 기반의 선언적 권한 모델을 제시함으로써, 보안성과 확장성이 결합된 기업용 에이전트 설계의 실용적 기반을 제공하는 데 기여한다.

Abstract

Recently, the Large Language Model (LLM)-based AI agent ecosystem is rapidly transitioning toward autonomous task automation systems, supported by the standardization of the Model Context Protocol (MCP). However, conventional agent framework studies have focused primarily on static single- or multi-agent pipeline designs. Consequently, they exhibit critical limitations in dynamically composing modules at runtime to meet changing business requirements, and they remain highly vulnerable to security breaches caused by excessive tool privileges. To address these limitations, this study proposes a secure and modular AI work agent architecture that adopts MCP as a standardized inter-module interface, allowing users to compose only the necessary functionalities dynamically. The proposed system is established on four software engineering design principles—Minimal Core, Declarative Module Registry, Least Privilege, and Human-in-the-Loop—each explicitly mapped to ISO/IEC 25010 software quality characteristics. The architectural validity and security compliance of the proposed system are demonstrated through comparative analysis with existing tools, mapping against prior security guidelines (OWASP, Databricks, AWS), and scenario-based walkthroughs. This study provides a practical, open-standard design foundation crucial for building secure, scalable, and compliant AI agents in modern industrial environments.

한글키워드 : AI 에이전트, 모듈형 아키텍처, Model Context Protocol(MCP), 최소 권한, Human-in-the-Loop, ISO/IEC 25010
keywords : AI Agent, Modular Architecture, Model Context Protocol(MCP), Least Privilege, Human-in-the-Loop, ISO/IEC25010

* 수원대학교 클라우드융복합
** 배재대학교 전기전자공학과
† 교신저자: 배선영(email: sypae@pcu.ac.kr)

접수일자: 2026.06.01. 심사완료: 2026.06.05.
게재확정: 2026.06.20.

1. 서 론

2024년 11월 Anthropic이 MCP(Model Context Protocol)를 공개하고 2025년 3월 OpenAI가 공식 채택하면서, AI 에이전트 생태계에 표준화된 도구 인터페이스가 본격적으로 자리를 잡았다. 2025년 12월에는 Linux Foundation 산하 Agentic AI Foundation(AAIF)으로 MCP가 이관되어 특정 벤더 종속에서 벗어난 중립적 표준으로 전환되었으며, 공개 이후 1년 만에 18,000개 이상의 MCP 서버가 구축될 정도로 급격한 생태계 성장이 이루어졌다[1]. 이 흐름은 단순한 챗봇 보조 도구에서 '실제로 일하는 AI'로의 패러다임 전환을 촉진하고 있다.

이러한 변화에도 불구하고, 현존하는 AI 에이전트 도구들은 두 가지 구조적 한계를 보이고 있다. 첫째, OpenClaw처럼 기능 집약적인 올인원 도구는 설치 복잡도가 높고 불필요한 권한을 일괄 부여하는 경향이 있다. 둘째, Claude Cowork처럼 사용이 간편한 도구는 제한적 확장성으로 인해 복잡한 사용자 정의 워크플로우를 구성하는 데 한계가 있다[2]. Wang et al.(2024)은 LLM 기반 자율 에이전트 연구에서 에이전트의 프로파일링·메모리·계획·행동 구성 요소가 유기적으로 통합되어야 실질적인 업무 자동화가 가능함을 지적하였다[3].

본 연구는 이 문제를 해소하기 위해 MCP를 모듈 인터페이스 표준으로 채택한 모듈형 AI 업무 에이전트 아키텍처를 제안한다. 핵심 기여는 네 가지다. (1) 최소 코어와 선택적 모듈의 분리 원칙을 통한 유연한 기능 조합 구조 제시, (2) 선언적 YAML 권한 모델과 ISO/IEC 25010 소프트웨어 품질 특성의 명시적 매핑을 통한 설계 타당성 논증, (3) Human-in-the-Loop의 계층적 적용(L0~L3 위험도 기반)을 통한 OWASP LLM08 과도한 에이전시 위협 차단 메커니즘 설계, (4) 악성 MCP 서버에 의한 공급망 공격(Tool Poisoning)에 대한 1차 방어 구조 논의이다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 검토하고, 3장에서 연구 문제 및 설계 방법을 제시한다. 4장에서 시스템 구현 내용을 기술하고, 5장에서 보안 설계 원칙을 상세화한다. 6장에서 설계 타당성을 다각도로 분석하며, 7장에서 결론을 제시한다.

2. 관련 연구

2.1 LLM 기반 자율 에이전트 연구

LLM 기반 에이전트는 추론·계획·도구 사용 능력을 결합한 자율 시스템으로, 단순 워크플로와 구별된다. Wang et al.(2024)은 에이전트 구성 요소를 프로파일링, 메모리, 계획, 행동의 4축으로 체계화하였으며[3], Li et al.(2024)은 LLM 기반 멀티에이전트 시스템의 워크플로·인프라·도전과제를 종합 분석하며 모듈화가 복잡한 에이전트 시스템의 유지보수성을 획기적으로 향상시킨다고 주장하였다[4].

Ferrag et al.(2025)은 2019~2025년간 발표된 60여 개 벤치마크와 에이전트 프레임워크를 분석하여, MCP·ACP(Agent Communication Protocol)·A2A(Agent-to-Agent Protocol) 등 에이전트 협력 프로토콜이 차세대 연구의 핵심 축으로 부상하고 있음을 확인하였다[5]. Zhang et al.(2024)의 AFlow 연구는 몬테카를로 트리 탐색(MCTS)을 활용하여 에이전트 워크플로를 자동 최적화함으로써, 소형 LLM이 대형 LLM을 능가하는 비용 효율적 구성이 가능함을 입증하였다[6].

2.2 모델 컨텍스트 프로토콜(MCP)

MCP는 JSON-RPC 2.0 기반의 전송 계층과

Tools·Resources·Prompts의 3가지 핵심 프리미티브로 구성되며, 호스트(Host)-클라이언트(Client)-서버(Server)의 3계층 아키텍처를 정의한다[7]. 이 구조는 본 논문이 제안하는 '사용자 인터페이스 레이어-에이전트 코어-모듈 레이어'의 3계층과 논리적으로 대응된다.

Ayyagari(2025)는 MCP가 HTTP · WebSocket · stdio 등 다양한 전송 메커니즘을 지원하는 맥락 인식(context-aware) 인터페이스 구조로, 기업 환경 에이전트 배포에 최적화되어 있다고 분석하였다[8]. Zhang et al.(2025)의 CA-MCP 연구는 기존 MCP의 무상태(stateless) 구조에 공유 컨텍스트 저장소(Shared Context Store)를 결합함으로써 다중 에이전트 간 협력 효율이 유의미하게 향상됨을 실험적으로 검증하였다[9]. Salesforce AI Research(2025)의 MCP-Universe 벤치마크는 6개 도메인 231개 실제 과제를 통해 MCP 환경에서 LLM의 도구 사용 능력을 종합 평가하였으며, 최고 성능 모델(GPT-5)도 43.72%의 성공률에 머물러 실무 적용 시 아키텍처 수준의 설계 보완이 필수적임을 시사하였다[10].

2.3 에이전트 보안

에이전트 보안 연구에서는 최소 권한 원칙(Least Privilege)과 HitL 메커니즘이 핵심 방어 수단이다. OWASP Agentic Security Initiative는 15개 위협 범주를 체계화하며, 이 중 LLM08 '과도한 에이전시(Excessive Agency)'를 AI 에이전트 고유의 최우선 위협으로 분류하고 HitL 체크포인트를 필수 통제 요소로 제시하였다[11]. Databricks DASF v3.0은 35개 에이전트 신규 위험과 최소 권한(DASF 5, 57, 64), HitL 감독(DASF 66), 샌드박싱(DASF 34, 62)의 6개 완화 통제를 제안하였다[12].

공급망 보안(Supply-chain Security) 측면에서, 2025년 4월 보안 연구자들이 발표한 분석에서는 MCP 서버가 악의적으로 조작된 도구 설명(Tool Description)을 통해 에이전트를 의도치 않은 행동으로 유도하는 'Tool Poisoning' 공격이 실질적 위협임이 확인되었다[4][13]. AWS(2026)는 에이전트 보안의 4대 원칙으로 전통적 사이버보안 기초, 최소 권한 접근 관리, 결정론적 외부 통제, 지속적 관찰가능성을 제시하며, 인프라 수준의 강제 통제가 프롬프트 기반 가이드레일보다 우선시되어야 한다고 권고하였다[14].

3. 연구 문제 및 설계 방법

3.1 연구 문제 정의

본 연구는 다음 세 가지 연구 문제를 설정한다.

- RQ1 : MCP를 모듈 인터페이스 표준으로 채택할 때, 에이전트의 기능 확장성과 LLM 호환성이 동시에 달성될 수 있는가?
- RQ2 : 선언적 YAML 권한 모델은 코드 분석 없이 모듈 신뢰성 검증을 가능하게 하는가?
- RQ3 : Human-in-the-Loop 메커니즘이 권한 위반 발생을 실효적으로 억제할 수 있는가?

3.2 기존 도구와의 비교 분석

표 1에서는 기존 방식인 OpenClaw와 Claude Cowork를 제안하는 시스템과 비교한 것이다. 접근 방식에서 경량 코어와 선택적 모듈만을 사용하는 접근 방식으로 표준 MCP 프로토콜을 사용하고 보안 모델에서는 최소한의 권한만을 허용하는 것으로 하며, L2 이상 수정 불가능한 기록(로그)으로 감사 추적함을 보인다.

3.3 4대 설계 원칙

제안 시스템의 설계는 다음 4개 원칙의 교차 적용을 통해 기능성·보안성·사용성의 균형을 나

타내었다.

표 1. 전통적인 AI 에이전트 도구 비교
Table 1. Comparison of Traditional AI Agent Tools

Category	OpenClaw	ClaudeCowork	Proposal System
Approach	Full-stack self-hosted	Desktop app (built-in)	Lightweight core + selective modules
Scope	All-in-one features	File & browser-centric	Activate only needed features
Module Interface	Proprietary skill/hook	Proprietary plugin	Standard MCP protocol
Permission Model	User-config dependent	App-level (opaque)	Declarative least-privilege (module.yaml)
Audit Trail	None	None	Immutable log for L2+
LLM Compatibility	Claude only	Claude only	All MCP-compatible LLMs

• 경량 코어 – 에이전트 루프·메모리·보안의 핵심만 코어에 두고, 나머지는 모두 모듈로 분리

• 선언적 모듈 – 모듈은 YAML 메타데이터로 기능·의존성·권한을 선언. 코드 없이 모듈 동작 범위 파악 가능

• 최소 권한 원칙 – 모듈은 선언한 권한 범위 밖의 작업 수행 불가. 파일 읽기 모듈은 파일 쓰기 불가

• Human-in-the-Loop – 파괴적 작업(삭제·외부 전송·결제)은 반드시 사용자 확인 절차를 거침

각 원칙은 국제 소프트웨어 품질 표준인 ISO/IEC 25010의 품질 특성과 명시적으로 대응되어 아키텍처 결정의 학술적 근거를 제공한다.

• 경량 코어 – 유지 관리성(모듈화). 코어를 기능과 분리하여 변경 전파를 최소화하고 독립적인 업데이트 가능

• 선언적 모듈 – 분석성/테스트성. YAML 메타데이터 사용으로 소스 코드를 검사하지 않고도 사전 실행 분석 및 모듈 동작의 정적 검증 가능

• 최소 권한 원칙 – 보안(기밀성/무결성). 무단 액세스 및 데이터 변조를 차단하며, 모듈 경계에서 제로 트러스트 원칙 구현

• Human-in-the-Loop – 사용성(사용자 오류 방지). 파괴적인 작업에 대한 명시적인 승인 게이트를 통해 에이전트 오작동 및 OWASP LLM08 과도한 에이전트 위협 방지

3.4 시스템 아키텍처

제안 아키텍처는 사용자 인터페이스 레이어, 에이전트 코어(Core), 모듈 레이어(Modules)의 3계층으로 그림 1과 같이 구성된다.

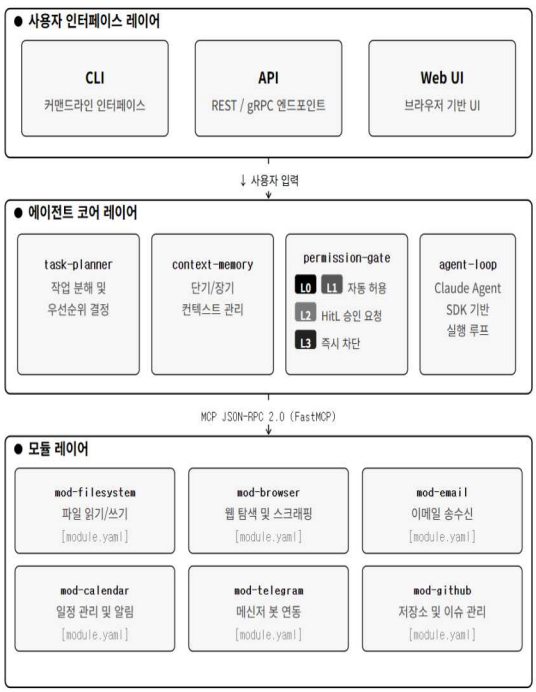

그림 1. 제안 시스템의 3계층 아키텍처 구조
Fig. 1. The three-tier architectural structure of the proposed system

이는 MCP 표준의 Host-Client-Server 3계층과 논리적으로 대응된다. 에이전트 코어는 LLM 추론 루프(Claude Agent SDK 기반), 세션 컨텍스트 메모리, 권한·보안 게이트, 작업 플래너의 4개 서브컴포넌트로 구성된다. 코어는 변경 없이 유지되며, 기능 확장은 전적으로 모듈 레이어에 의해 이루어진다. 각 모듈은 FastMCP 라이브러리를 사용하여 MCP 서버 형태로 구현되며, 코어는 표준 MCP 프로토콜을 통해 모듈의 도구(tool)를 호출한다. MCP 표준 프로토콜이 코어와 모듈 레이어의 결합을 담당하며, 코어는 변경 없이 모듈 단위 기능 확장이 가능하다.

3.5 모듈 메타데이터 규격(module.yaml)

모든 모듈은 선언적 권한 모델을 구현하는 module.yaml 파일을 포함해야 한다.

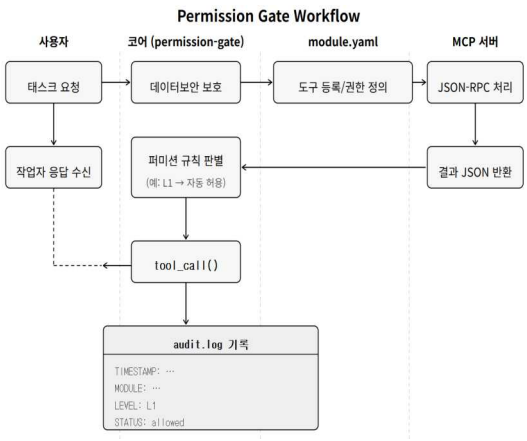

그림 2. module.yaml 기반 권한 검사와 MCP 도구 호출 시퀀스
Fig. 2. module.yaml-based permission check and MCP tool call sequence

이 파일은 ① 접근 가능한 네트워크 도메인, ② 파일 시스템 경로, ③ 외부 서비스, ④ 필요 환경변수, ⑤ 노출 도구 목록, ⑥ 위험도 등급(low/medium/high), ⑦ 샌드박스 사용 여부를 명시한다. 선언적 규격은 코드를 읽지 않고도 모듈의 동작 범위를 파악할 수 있게 하여 RQ2가 제기한 '코드 분석 없는 신뢰성 검증' 문제에 구조적으로 응답한다.

권한 게이트는 module.yaml의 선언적 메타데이터를 런타임에 조회하여 코드 수정 없이 모듈 동작 범위를 통제한다. 그림 2는 module.yaml 기반 권한 검사와 MCP 도구 호출 시퀀스이다.

4. 시스템 구현

4.1 기술 스택

제안 시스템의 기술 스택은 에이전트 런타임으로 Claude Agent SDK(Python/TypeScript), 모듈 통신 인터페이스로 MCP(JSON-RPC 2.0), 모듈 서버 구현으로 FastMCP, 설정 관리로 YAML, 선택적 보안 샌드박스로 Docker를 사용하였다. 모듈 관리 CLI는 npm/pip 패키지 생태계를 활용하여 모듈 검색·설치·업데이트를 지원한다.

4.2 에이전트 코어 구현

코어의 ModularAgent 클래스는 초기화 시 config.yaml에서 활성화된 모듈 목록을 읽어 ModuleRegistry에 로드하고, PermissionGate에 각 모듈의 권한을 등록한다. 에이전트 실행 시 활성화된 모듈의 MCP 서버만 허용 도구로 등록하여 Claude Agent SDK의 query 함수에 전달한다. 권한 게이트는 모듈이 module.yaml에 선언하지 않은 도메인·경로에 접근을 시도할 경우 즉시 차단하고 audit.log에 기록한다.

에이전트 루프의 실행 흐름은 Figure 1과 같이 입력 수신, 태스크 분해, 권한 검사, 도구 호출, 추론, 메모리 갱신의 6단계로 구성된다. 각 단계

의 구체적 처리 과정은 다음과 같다.

(1) 입력 수신 및 전처리 : CLI 또는 API를 통해 수신된 사용자 입력은 UTF-8로 정규화된 후 세션 컨텍스트 메모리에 저장된다. 이전 대화 맥락이 존재할 경우 context-memory 컴포넌트가 관련 기록을 검색하여 현재 요청과 함께 LLM에 전달한다.

(2) 태스크 분해 : task-planner는 LLM을 호출하여 단일 사용자 요청을 실행 가능한 원자적(atomic) 단계의 순서열로 변환한다. 각 단계에는 대상 모듈 ID, 호출할 MCP 도구 이름, 인수(arguments), 예상 위험 등급(L0~L3)이 명시된다.

(3) 권한 검사 : permission-gate는 각 단계의 위험 등급을 판별하여 L0·L1 단계는 자동 허용하고, L2 단계는 HitL 프롬프트를 통해 사용자 승인을 대기하며, L3 단계는 즉시 차단하고 audit.log에 기록한다. 이 검사는 MCP 도구 호출 직전에 동기적으로 수행되어 권한 우회를 원천적으로 방지한다.

(4) MCP 도구 호출 : Claude Agent SDK의 tool_use 메커니즘을 통해 활성화된 모듈의 MCP 서버에 JSON-RPC 2.0 형식으로 요청이 전달된다. 모듈은 module.yaml에 선언된 범위 내의 리소스에만 접근하며, 응답은 구조화된 JSON으로 반환된다.

(5) 추론 및 합성 : LLM은 도구 호출 결과와 컨텍스트 메모리를 결합하여 사용자에게 전달할 자연어 응답을 생성한다. 도구 호출 실패 시에는 재시도 없이 오류 사유를 명시한 응답을 반환하여 무한 루프를 방지한다.

(6) 메모리 갱신 : 태스크 완료 후 요약된 실행 결과가 context-memory에 저장되어 후속 대화에서 참조 가능하다. L2 이상의 모든 액션은 타임스탬프·세션 ID와 함께 audit.log에 불변 기록된다.

이러한 루프 구조는 Wang et al.(2024)[3]이 제시한 '프로파일링·메모리·계획·행동'의 4축 에이전트 구성 요소를 본 시스템의 설계 원칙과 대응시킨 것으로, 경량 코어와 선언적 모듈 분리를 통해 실질적인 유지보수성을 확보한다.

4.3 모듈 구현: mod-telegram 사례

mod-telegram은 FastMCP 라이브러리를 사용하여 MCP 서버 형태로 구현한다. send_message 도구는 지정된 Telegram 채팅방에 Markdown 형식의 메시지를 발송하며, get_recent_messages 도구는 최근 수신 메시지를 조회한다. 두 도구 모두 module.yaml에서 network: ["api.telegram.org"]를 유일한 허용 도메인으로 선언하며, 파일 시스템 접근 권한은 갖지 않는다. 위험도는 'low'로 설정되어 L0·L1 수준 작업은 자동 허용된다.

4.4 표준 모듈 카탈로그

코어 번들은 agent-loop, context-memory, permission-gate, task-planner의 4개 컴포넌트로 구성된다. 공식 선택 모듈은 표 2와 같다.

4.5 복합 태스크의 서브에이전트 위임 패턴

단일 태스크가 복수의 독립 모듈을 요구하는 경우, task-planner는 의존 관계 분석을 통해 병렬 실행 가능한 단계를 식별하고 팬아웃(Fan-out) 패턴을 적용한다. 이 패턴은 다음 세 단계로 동작한다.

단계 1 : DAG 생성 (Directed Acyclic Graph)

task-planner가 각 단계 간 데이터 의존성을 분석하여 실행 순서 그래프를 생성한다. 예를 들어, '주간 보고서 작성' 요청에서 mod-calendar(일정 수집)와 mod-email(메일 수집)은 상호 의존성이 없으므로 병렬 노드로 배치되고, 이후 LLM 합성 단계가 단일 수렴 노드로 연결된다.

표 2. 공식 모듈 카탈로그
Table 2. Official Module Catalogue

Module	Key Functions	Dependency	Risk Level
mod-filesystem	Read/write/organise/ search files	None	Medium (L1–L2)
mod-browser	Web navigation, form input, scraping (CDP)	Playwright	Medium (L2)
mod-email	Read/send/classify emails	Gmail/Outlook OAuth	High (L2–L3)
mod-calendar	Query/create/edit calendar events	Google Calendar OAuth	Medium (L1)
mod-telegram	Receive/send Telegram bot messages	Bot API Token	Low (L1)
mod-github	Monitor PRs, issues, and commits	GitHub PAT	Medium (L1–L2)

단계 2 : 권한 일괄 검사

병렬 실행 전에 permission-gate는 모든 노드의 위험 등급을 일괄 평가한다. 하나의 노드라도 L2 이상이면 전체 계획을 사용자에게 제시하고 단일 승인으로 처리하여, 다중 확인 요청으로 인한 사용성 저하를 방지한다.

단계 3 : 결과 수렴 및 합성

병렬 MCP 호출이 완료된 후 각 결과가 context-memory에 집약되며, LLM이 통합 추론을 수행한다. 개별 노드 실패 시 해당 노드의 결과를 'unavailable'로 표기하고 나머지 결과만으로 부분 응답을 생성하여 전체 태스크 실패를 방지한다.

이 패턴은 Li et al.(2024)[4]이 지적한 다중 에이전트 시스템의 조율 복잡도 문제를, 외부 오케스트레이터 없이 단일 코어 내 DAG 스케줄링으로 해결하는 경량 접근법이다. 향후 CA-MCP[9]의 공유 컨텍스트 저장소와 결합할 경우 에이전트 간 협업 효율을 추가로 향상시킬 수 있다.

5. 보안 설계 원칙

5.1 4계층 권한 모델과 OWASP LLM08 대응

제안된 시스템은 에이전트 작업을 위험도에 따라 L0~L3의 4계층으로 분류하였다. 이 모델은 OWASP LLM08 '과도한 에이전시(Excessive Agency)' 위협에 직접 대응하는 설계 전략으로 상세 설명은 다음과 같다.

- L0 – 읽기 전용(Read-only). 파일을 읽거나 데이터를 조회하는 가장 낮은 단계 권한. 파일 목록 확인이나 캘린더 읽기 등이 해당하며, 시스템은 이를 자동 승인(Auto-allow)함
- L1 – 쓰기(Write). 파일을 생성, 수정하거나 메시지를 보내는 등 데이터를 변경하는 권한. 텔레그램 메시지 전송이나 파일 저장이 예시이며, 자동 승인하되 로그를 기록(Auto-allow + log)하여 추후 추적이 가능하게 함
- L2 – 파괴적 작업(Destructive). 파일 삭제나 이메일 발송처럼 시스템이나 데이터에 영구적인 영향을 줄 수 있는 권한. 이 단계에서는 반드시 사람의 승인(HitL: Human-in-the-Loop approval required)이 있어야 작업이 수행됨
- L3 – 금융/핵심 작업(Financial/Critical). 결제나 예약 등 금전적 손실이나 중대한 문제를 일으킬 수 있는 가장 높은 단계의 권한. 항공권 예약이나 물품 구매 등이 포함되며, 기본적으로 항상 차단(Always blocked)된 상태를 유지함

L2 이상의 파괴적 액션은 반드시 사용자 승인을 거치며, L3(금융/핵심)는 항상 차단된다. 그림 3은 위험 등급(L0~L3) 기반 Human-in-Loop 결

정 흐름도이다.

5.2 공급망 보안: Tool Poisoning 방어

2025년 4월 보안 연구자들이 발표한 분석에서 악성 MCP 서버가 도구 설명(Tool Description)을 조작하여 에이전트를 의도치 않은 행동으로 유도하는 'Tool Poisoning' 공격이 실질적 위협으로 확인되었다[4][13].

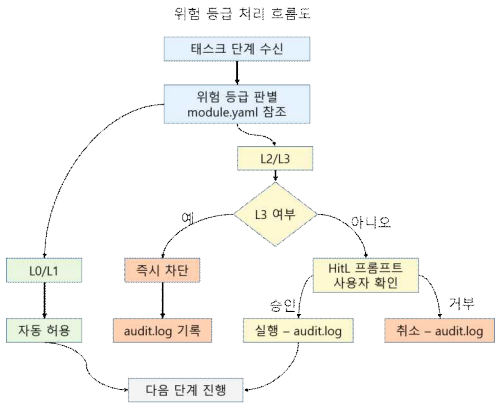

그림 3. 위험 등급(L0~L3) 기반 Human-in-the-Loop 결정 흐름도
Fig. 3. Human-in-the-Loop decision flow chart based on risk class (L0 to L3)

제안 시스템의 선언적 module.yaml 권한 모델은 이에 대한 1차 방어선(First Line of Defence) 역할을 수행한다. 구체적으로, ① 모듈이 module.yaml에 선언한 도메인 외의 네트워크 호출을 인프라 레벨에서 차단하고, ② 커뮤니티 모듈 설치 시 자동화 파이프라인이 module.yaml의 권한 선언을 정적 분석하여 과도한 권한 요청을 플래그 처리하며, ③ 사용자에게 권한 목록을 설치 전 명시적으로 노출함으로써 사회공학적 기만을 억제한다. 다만, Tool Description 자체의 의미론적 조작에 대한 방어는 선언적 모델만으로 완전하지 않다. 이는 향후 도구 설명 보강(Augmented Tool Description)[4] 및 런타임 행동 모니터링과의 결합을 통해 보완할 필요가 있다.

5.3 보안 설계 요소별 원칙적 적용

제안 시스템의 보안 설계는 다음 원칙을 준수한다. ① 네트워크 접근 통제: module.yaml 선언 도메인 외 접근 차단. ② 파일 시스템 통제: allowed_dirs 범위 외 파일 접근 차단. ③ 자격증명 보호: API 키 환경변수 주입, 설정 파일 평문 저장 금지. ④ 고위험 모듈 격리: 위험도 'high' 모듈은 Docker 격리 권장. ⑤ 감사 추적: L2 이상 작업은 타임스탬프·세션 ID와 함께 audit.log에 불변 기록. 이는 AWS가 강조한 '결정론적 외부 통제'[14] 원칙을 구조적으로 구현한 것이다.

6. 설계 타당성 분석

6.1 보안 권고 매핑 분석

제안 시스템은 첫째 기능 조합 유연성 측면에서 MCP 표준 프로토콜 채택으로 모듈을 런타임에 선택적으로 활성화·비활성화할 수 있으며, 이는 Ayyagari(2025)[8]가 지적한 MCP의 '모듈형 클라이언트-서버 구조'의 실용적 구현이다. 둘째, 권한 가시성 측면에서 module.yaml의 선언적 권한 모델은 비개발자도 모듈의 접근 범위를 즉시 파악할 수 있어 OWASP ASI[11]가 강조하는 '에이전트 행동 범위의 명확한 경계 설정' 원칙을 구조적으로 실현한다. 마지막으로 공급망 보안 측면에서 기존 도구들이 커뮤니티 모듈에 대한 체계적 권한 검토 구조를 갖추지 못한 반면, 제안 시스템의 module.yaml은 Tool Poisoning에 대한 1차 방어선을 제공한다. 표 3은 제안 시스템의 각 설계 요소가 선행 연구 및 산업 보안 권고와 어떻게 대응되는지를 정리한 것이다.

6.2 시나리오 기반 검토

구체적인 실험 데이터 없이도 시스템의 가동성(Operability)을 입증하기 위해, 세 가지 시나리오에서 데이터가 코어와 모듈 사이를 흐르는 과정을 단계별로 기술한다.

표 3. 디자인 요소와 보안 지침의 매핑
Table 3. Mapping of Design Elements to Security Guidelines

Proposed Design Element	Security Guideline / Standard	Reference
module.yaml declarative permission	Least Privilege per module scope	11,12
L2 HitL approval gate	HitL for high-risk actions (DASF 66); OWASP LLM08 Excessive Agency	12,14
Docker sandbox isolation	Sandboxing & Isolation (DASF 34, 62)	12
allowed_dirs file access control	Deterministic external controls	14
audit.log immutable record	Continuous observability & audit trail	11, 14
MCP standard protocol	Open standard for agentic interoperability (AAIF, Linux Foundation)	5, 8

【시나리오 A : 아침 브리핑 – 복합 모듈 파이프라인】 ① 사용자가 CLI에서 '오늘 일정 요약하고 Telegram으로 보내줘'를 입력한다. ② 에이전트 코어의 task-planner가 'calendar.query → telegram.send' 2단계 계획을 수립한다. ③ permission-gate가 mod-calendar(L0 읽기)·mod-telegram(L1 쓰기) 권한을 각각 자동 허용한다. ④ mod-calendar가 Google Calendar OAuth API를 통해 당일 일정을 조회하고 구조화된 JSON으로 반환한다. ⑤ LLM 루프가 JSON을 자연어 요약으로 변환한다. ⑥ mod-telegram이 api.telegram.org에만 요청하여 요약 메시지를 발송하고, 이 과정에서 파일 시스템에는 일절 접근하지 않는다. 이 흐름은 '최소 권한 원칙'이 모듈 파이프라인 전체에 전파됨을 보여준다.

【시나리오 B : 문서 정리 – HitL 게이트 동작】 ① 사용자가 'Downloads 폴더 정리해줘'를 입력한다. ② mod-filesystem이 allowed_dirs 범위 내에서 파일 목록(47개 PDF, 23개 이미지, 31개 기타)을 조회한다(L0 자동 허용). ③ task-planner가 파일 이동 계획을 수립하고 L2(파괴적) 작업으로 분류한다. ④ permission-gate가 HitL 게이트를 발동하여 계획을 사용자에게 제시하고 명시적 승인을 요청한다. ⑤ 사용자 승인 후에만 파일 이동이 실행되며, 전체 과정이 audit.log에 기록된다. ⑥ 사용자가 취소할 경우 어떤 파일도 이동되지 않는다. 이 흐름은 OWASP LLM08 위협을 HitL이 실제로 차단하는 과정을 명확히 보여준다.

【시나리오 C : 악성 MCP 서버 공급망 공격 방어】 ① 공격자가 조작된 tool description을 포함한 mod-analytics를 커뮤니티 레지스트리에 배포한다. ② 사용자가 mod-analytics 설치를 시도하면, 검증 파이프라인이 module.yaml을 정적 분석하여 network: ["exfil.attacker.com"]이라는 비정상 도메인 선언을 감지하고 경고를 표시한다. ③ 설치 후에도 권한 게이트가 api.telegram.org 이외의 네트워크 호출을 인프라 레벨에서 차단하여 데이터 유출을 방지한다. 이 시나리오는 선언적 권한 모델이 Tool Poisoning에 대한 1차 방어선으로 작동함을 보여준다.

6.3 설계 한계 및 향후 실증 평가 프레임워크

본 연구는 아키텍처 설계 제안과 정성적 분석

에 초점을 맞추고 있으며, 다음과 같은 한계를 명시적으로 갖고 있다. 첫째, 실제 프로토타입 구현을 통한 정량적 성능 측정이 수행되지 않았다. 둘째, 다수 사용자를 대상으로 한 SUS 기반 사용성 평가가 포함되지 않았다. 셋째, Tool Poisoning 방어 실효성에 대한 실험적 검증이 이루어지지 않았다. 이러한 한계는 향후 연구에서 진행될 예정이며, 향후 실증 평가에서 사용할 핵심 성과 지표(KPI)는 다음과 같이 정의될 수 있다.

- 효율성(Efficiency) - 지표 : 평균 작업 지연 시간(Mean Task Latency, ms). 내용: 사용자의 질의부터 최종 결과가 나올 때까지의 시간 측정. 이를 통해 MCP 프로토콜의 오버헤드가 적절한지 검증
- 정확성(Accuracy) - 지표 : 도구 선택 F1-스코어(Tool Selection F1-Score). 내용: 계획 단계에서 MCP 모듈을 얼마나 정확하게 호출했는지 측정. 정밀도(Precision)와 재현율(Recall)을 종합하여 도구 선택의 정확도를 평가
- 안전성(Safety) - 지표 : 미승인 차단율(Unauthorised Block Rate, %). 내용: 보안 게이트가 권한 밖의 행동을 성공적으로 차단한 비율 측정. 시스템이 안전 가이드라인을 얼마나 잘 준수하는지 확인
- 확장성(Extensibility) - 지표 : 모듈 추가 시간(Module Addition Time, min). 내용: 새로운 MCP 모듈을 설치, 등록, 활성화하는 데 걸리는 전체 시간 측정
- 사용성(Usability) - 지표 : SUS 점수(System Usability Scale, 0 - 100). 내용: 시나리오 기반 과업을 마친 최종 사용자들을 대상으로 설문하여 시스템의 편리함을 0~100점 사이의 점수로 측정

6.3.1 KPI 측정 방법론

위에서 정의한 5개 KPI의 실증 평가를 위해 다음과 같은 측정 방법론을 설계한다.

(1) 효율성 – 평균 태스크 지연 시간(Mean Task Latency) : audit.log에 기록된 task_start 및 task_complete 이벤트의 타임스탬프 차이를 자동 집계한다. MCP 프로토콜 오버헤드 분리를 위해 총 지연 시간을 LLM 추론 시간(T_llm), MCP 직렬화 시간(T_mcp), 네트워크 왕복 시간(T_net)으로 분해 측정한다. 기준선(Baseline)으로는 MCP 미사용 직접 API 호출 방식의 지연 시간을 사용한다.

(2) 정확성 – 도구 선택 F1-스코어(Tool Selection F1-Score) : 100개의 표준 시나리오에 대해 정답 모듈 호출 시퀀스를 사전 정의한 골든셋(Golden Set)을 구성한다. task-planner가 생성한 계획과 골든셋을 비교하여 정밀도(Precision)와 재현율(Recall)을 산출한다. 불필요한 모듈 호출(False Positive)과 누락된 호출(False Negative)이 각각 보안 위험과 기능 손실로 이어짐을 감안하여 F1 가중치를 동등하게 설정한다.

(3) 안전성 – 미승인 차단율(Unauthorised Block Rate) : permission-gate의 차단 이벤트를 audit.log에서 추출하고, 의도적으로 권한 범위를 초과하는 100개의 악의적 입력 시나리오에 대한 차단 성공률을 측정한다. 시나리오 B(공격 방어)의 경우 프롬프트 인젝션을 통한 모듈 권한 확장 시도를 포함하여 L0~L3 각 등급별 차단율을 분리 측정한다.

(4) 확장성 – 모듈 추가 시간(Module Addition Time) : 신규 MCP 모듈의 module.yaml 작성 완료 시점부터 CLI install 명령, 모듈 레지스트리 등록, 첫 번째 도구 호출 성공까지의 전체 소요 시간을 측정한다. 비교 기준으로 OpenClaw의 플러그인 추가 절차와 소요 시간을 병행 측정한다.

(5) 사용성 – SUS 점수(System Usability

Scale) : 시나리오 A(복합 업무)와 시나리오 B(문서 정리) 완료 후 비개발자 5명, 개발자 5명을 대상으로 표준 SUS 10문항 설문을 수행한다. SUS 점수 70점 이상을 수용 가능한 사용성 기준으로 설정하며, 각 사용자 그룹별 점수 차이를 분석하여 모듈 추상화의 일반 사용자 접근성을 평가한다.

실험 시나리오는 두 가지로 설계할 예정인데, 시나리오 α(복합 업무)는 캘린더·이메일·파일 시스템 모듈이 협업하여 주간 보고서를 작성하는 프로세스의 성공률을 측정하고, 시나리오 β(공격 방어)는 간접 프롬프트 인젝션을 통해 모듈 권한 확장을 시도할 때 선언적 권한 모델의 비인가 차단율을 측정한다. 이를 통해 RQ1~RQ3에 대한 정량적 응답을 제공할 예정이다.

7. 결론

본 연구는 MCP를 모듈 인터페이스 표준으로 채택한 모듈형 AI 업무 에이전트 아키텍처를 제안하고, 설계 관점 비교·보안 권고 매핑·시나리오 기반 보행 분석을 통해 아키텍처의 타당성을 다각도로 논증하였다. 주요 결론은 다음과 같다.

첫째, MCP 기반 모듈형 아키텍처는 기능 확장성과 LLM 호환성을 동시에 달성하는 실용적 설계 방법론이다. 2025년 12월 AAIF로의 이관으로 MCP의 장기 중립성이 보장됨에 따라[1], 제안 시스템의 모듈 생태계는 벤더 종속 없이 지속 가능한 확장이 가능하다. 둘째, 선언적 YAML 권한 모델은 ISO/IEC 25010의 분석성·시험성 품질 특성을 구조적으로 실현하며, Tool Poisoning 공급망 위협에 대한 1차 방어선으로 기능한다. 셋째, L0~L3 계층 기반 HitL은 OWASP LLM08 과도한 에이전시 위협을 아키텍처 수준에서 차단하는 효과적인 메커니즘이다. 넷째, A2A(Agent-to-Agent) 프로토콜 등 AAIF에서 논의 중인 차세대 표준과의 연동 가능성이 제안 아키텍처의 MCP 표준 채택으로 인해 구조적으로 열려 있어, 본 연구의 시의성(Timeliness)이 높다.

향후 연구 방향으로는 ① 프로토타입 구현 및 KPI 프레임워크 기반 실사용자 대상 정량 실증 평가, ② CA-MCP[9] 방식의 공유 컨텍스트 저장소 통합을 통한 멀티에이전트 협력 효율 향상, ③ Augmented Tool Description[4]와 결합한 Tool Poisoning 방어 실효성 실험, ④ A2A 프로토콜과의 연동 설계 및 에이전트 간 신뢰 체계 연구를 제시한다.

참 고 문 헌

[1] Databricks, "Agentic AI Security: New Risks and Controls in the Databricks AI Security Framework (DASF v3.0)", 2025.

[2] Anthropic, "Building Effective Agents", 2025, https://www.anthropic.com/engineering/building-effective-agents

[3] M. A. Jayanti, X. Y. Han, "Enhancing MCP with Context-Aware Server Collaboration (CA-MCP)", arXiv:2601.11595v2, 2026, DOI: https://doi.org/10.48550/arXiv.2601.11595

[4] M. M. Hasan, H. Li, G. K. Rajbahadur, B. Adams, A. E. Hassan, "MCP Tool Descriptions Are Smelly! Towards Improving AI Agent Efficiency", arXiv:2602.14878, 2026, DOI: https://doi.org/10.48550/arXiv.2602.14878

[5] B. Liu, X. Li, J. Zhang, J. Wang, T. He, S. Hong, H. Liu, S. Zhang, K. Song, K. Zhu, Y. Cheng, S. Wang, X. Wang, Y. Luo, H. Jin, P. Zhang, O. Liu, J. Chen, H. Zhang, Z. Yu, H.

Shi, B. Li, D. Wu, F. Teng, X. Jia, J. Xu, J. Xiang, Y. Lin, T. Liu, T. Liu, Y. Su, H. Sun, G. Berseth, J. Nie, I. Foster, L. Ward, Q. Wu, Y. Gu, M. Zhuge, X. Tang, H. Wang, J. You, C. Wang, J. Pei, Q. Yang, X. Qi, C. Wu, "Advances and Challenges in Foundation Agents", arXiv:2504.01990, 2025, DOI: 10.48550/arXiv.2504.01990

[6] V. de Lamo Castrillo, H. K. Gidey, A. Lenz, A. Knoll, "Fundamentals of Building Autonomous LLM Agents" , arXiv:2510.09244, 2025, DOI: https://doi.org/10.48550/arXiv.2510.09244

[7] Jon Ander Oribe, "The Model Context Protocol (MCP): Emergence, Technical Architecture, and the Future of Agentic AI Infrastructure", ResearchGate, 2025, DOI: https://doi.org/10.5281/zenodo.17390299

[8] M. A. Ferrag, N. Tihanyi, M. Debbah, "From LLM Reasoning to Autonomous AI Agents: A Comprehensive Review", arXiv:2504.19678, 2025, DOI: https://doi.org/10.48550/arXiv.2504.19678

[9] X. Li, S. Wang, S. Zeng, Y. Wu, Y. Yang,"A Survey on LLM-Based Multi-Agent Systems: Workflow, Infrastructure, and Challenges", Vicinagearth, 1(1), p.9, 2024, DOI: https://doi.org/10.1007/s44336-024-00009-2

[10] M. Becattini, R. Verdecchia, E. Vicario, "SALLMA: A Software Architecture for LLM-based Multi-Agent Systems", Proceedings of IEEE/ACM SATrends, pp.5-8, 2025, DOI:https://doi.org/10.1109/SATrends66715.2025.00006

[11] OWASP Agentic Security Initiative, "AI Agent Security Cheat Sheet", 2025, https://cheatsheetseries.owasp.org, (Accessed Apr. 10, 2026)

[12] Z. Luo, Z. Shen, W. Yang, Z. Zhao, P. Jwalapuram, A. Saha, D. Sahoo, S. Savarese, C. Xiong, J. Li (Salesforce AI Research), "MCP-Universe: A Comprehensive Benchmark for Evaluating LLMs with Real-World MCP Servers", arXiv:2508.14704, 2025, DOI: https://doi.org/10.48550/arXiv.2508.14704

[13] Xinyi Hou, Yanjie Zhao, Shenao Wang, Haoyu Wang, "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions", arXiv:2503.23278, 2025, DOI: https://doi.org/10.48550/arXiv.2503.23278

[14] J. Zhang, J. Xiang, Z. Yu, F. Teng, X. Chen, J. Chen, M. Zhuge, X. Cheng, S. Hong, J. Wang, B. Zheng, B. Liu, Y. Luo, C. Wu, "AFlow: Automating Agentic Workflow Generation," in The Thirteenth International Conference on Learning Representations (ICLR 2025), 2025. https://openreview.net/forum?id=z5uVAKwmjf(arXiv preprint: arXiv:2410.10762, 2024)

저 자 소 개

송재숙(Jea-Suk Song)

1993.2 한남대학교 전자계산공학과 졸업
1995.2 한남대학교 전자계산공학과 석사
2007.2 한남대학교 컴퓨터공학과 박사
2025.8-현재 미르대표
2025.8-현재 수원대학교 겸임교수
<주관심분야> AI, BigData, IoT, LLM

배선영(Sun-Young Bae

2000.8 배재대학교 응용수학과 졸업
2004.2 충남대학교 정보통신공학과 석사
2006.2 충남대학교 정보통신공학과 박사수료
2015.3~현재 배재대학교 전기전자공학과 부교수
<주요관심분야> AI, 딥러닝, 빅데이터, IoT, 자동화시스템