Cuando empezamos a trabajar con agentes de inteligencia artificial, es normal que aparezcan conceptos que parecen hacer lo mismo pero que, en realidad, resuelven problemas distintos. Dos de los más fáciles de confundir son **MCP server** y **Skill**.

La diferencia esencial es esta:

- un MCP server le da **capacidades externas** al agente.
- una Skill le da una **forma concreta de hacer una tarea**.

Dicho de otra manera: el MCP server son las manos del agente. La Skill es el manual de trabajo.

Qué es un MCP server

Un **MCP server** es un servidor que expone herramientas, recursos o prompts a un agente de IA usando el protocolo MCP, que significa **Model Context Protocol**. La idea de MCP es sencilla: en vez de que cada aplicación de IA invente su propia forma de conectarse a GitHub, bases de datos, sistemas de ficheros, servidores SSH, APIs internas o herramientas de seguridad, usamos un protocolo común. Un MCP server puede exponer, por ejemplo, herramientas como estas:

```
ssh.exec
github.create_issue
postgres.query
wazuh.search_alerts
opensearch.query
filesystem.read_file
nmap.scan
```

Esto no significa que el modelo “sepa” auditar un servidor, explotar una máquina de CTF o generar un informe profesional. Significa que ahora tiene acceso a herramientas con las que puede obtener información o ejecutar acciones. Por eso, cuando hablamos de MCP, hablamos sobre todo de **conectividad**, **acciones** y **acceso controlado a sistemas externos**.

Un MCP server puede correr como un **proceso local** usando stdio, o puede **exponerse mediante transporte HTTP**, dependiendo del caso de uso. En entornos locales o de desarrollo, stdio puede ser suficiente. En entornos distribuidos, multiusuario o productivos, HTTP suele tener más sentido.

Qué es una Skill

Una **Skill** es un paquete de instrucciones, recursos, scripts y procedimientos reutilizables que le explica al agente cómo debe realizar una tarea concreta. Normalmente una Skill se organiza como una carpeta con un archivo principal llamado **SKILL.md**. Ese archivo describe cuándo debe usarse la skill, qué objetivo tiene, qué pasos debe seguir el agente, qué criterios debe aplicar y qué formato debe tener la salida.

Una Skill puede incluir:

- Instrucciones detalladas.
- Plantillas de informes.
- Criterios de severidad.
- Scripts auxiliares.
- Recursos de referencia.
- Ejemplos de entrada y salida.
- Flujos de trabajo repetibles.

La Skill no sustituye a una herramienta. Lo que hace es evitar que el agente improvise cada vez. Le damos una metodología. Por ejemplo, una Skill para generar informes de seguridad podría definir:

- Qué secciones debe tener el informe.
- Cómo clasificar vulnerabilidades.
- Qué evidencias debemos guardar.
- Cómo redactar el resumen ejecutivo.
- Qué tono usar para el cliente.
- Qué mitigaciones recomendar.
- Qué comprobaciones no deben faltar.

Sin esa Skill, el agente puede generar un informe, sí. Pero probablemente lo hará de forma variable, dependiendo del prompt, del contexto disponible y de cómo interprete la tarea en ese momento.

La diferencia práctica

La diferencia entre MCP server y Skill se entiende muy bien con esta comparación:

ASPECTO	MCP SERVER	SKILL
Qué aporta	Herramientas, recursos, prompts y acceso a sistemas externos	Procedimiento, metodología, instrucciones y recursos reutilizables
Para qué sirve	Para que el agente pueda hacer cosas fuera del modelo	Para que el agente haga una tarea de una forma concreta
Ejemplo	Ejecutar comandos por SSH, consultar una API, leer una base de datos	Auditar Debian siguiendo nuestra metodología
Dónde está la lógica	En el servidor MCP y en sus herramientas	En las instrucciones, scripts y recursos de la skill
Qué problema resuelve	Conexión y ejecución	Estandarización y repetibilidad
Riesgo principal	Acceso excesivo a sistemas reales	Instrucciones o scripts maliciosos o mal diseñados

Un resumen muy directo sería este:

```
MCP server = qué puede tocar el agente
Skill = cómo debe trabajar el agente
```

Ejemplo: generar un informe de seguridad

Supongamos que queremos que un agente genere un informe de seguridad de un servidor Debian. Si solo tenemos un MCP server, podemos darle herramientas como SSH, Nmap, Wazuh, OpenSearch o acceso al sistema de ficheros. El agente podrá obtener datos reales, ejecutar comandos y consultar evidencias. Pero no necesariamente sabrá qué estructura debe tener el informe, cómo priorizar hallazgos o qué criterios usar.

Si solo tenemos una Skill, podemos darle una metodología excelente: qué comprobar, en qué orden, qué plantilla usar, cómo redactar el resumen ejecutivo y cómo valorar la severidad. Pero si el agente no tiene herramientas para conectarse al servidor, no podrá obtener datos reales.

La combinación útil es usar las dos cosas:

```
Skill de auditoría Debian
  ├── Define metodología
  ├── Define plantilla
  ├── Define criterios de severidad
```

```
  ├── Define evidencias necesarias
  └── Usa herramientas expuestas por MCP

MCP servers
  ├── ssh.exec
  ├── filesystem.read
  ├── wazuh.search_alerts
  ├── opensearch.query
  ├── nmap.scan
  └── report.create
```

El resultado es mucho mejor: el agente tiene herramientas para obtener información y, al mismo tiempo, tiene un procedimiento claro para convertir esa información en un informe consistente.

Aplicado a ciberseguridad

En ciberseguridad, esta separación es especialmente importante. Un MCP server puede dar acceso a herramientas como:

- Nmap.
- Nuclei.
- Wazuh.
- OpenSearch.
- Suricata.
- GitHub.
- GitLab.
- PostgreSQL.
- SSH.
- Sistemas de ficheros.
- Herramientas OSINT.

…pero una Skill puede definir cómo debemos usar todo eso.

Por ejemplo, una Skill para pentesting web podría indicar que:

- Primero debemos identificar tecnologías.
- Después enumerar rutas.
- Luego analizar autenticación.
- Después probar controles de acceso.
- Luego buscar inyecciones.
- Después revisar subida de ficheros.
- Después validar impacto.
- Finalmente redactar hallazgos explotables con evidencia.

Sin esa Skill, el agente puede acabar ejecutando herramientas sin criterio. Con la Skill, el agente trabaja con una metodología más parecida a la de un operador humano.

Cuándo usar un MCP server

Nos conviene crear o usar un MCP server cuando necesitamos que el agente acceda a algo que no puede tocar por sí solo. Algunos casos claros:

- Conectarse a una base de datos.
- Ejecutar comandos en un servidor.
- Consultar una API externa.
- Leer documentación interna.
- Crear tickets.
- Consultar alertas de un SIEM.
- Lanzar escaneos de seguridad.
- Acceder a repositorios.
- Leer o escribir ficheros en un entorno controlado.

La pregunta que debemos hacernos es: **¿El agente necesita acceder a un sistema, dato o acción externa?** Si la respuesta es sí, probablemente necesitamos MCP o alguna herramienta equivalente.

Cuándo usar una Skill

Nos conviene crear una Skill cuando queremos que el agente haga una tarea siguiendo una metodología concreta. Algunos casos claros:

- Generar informes siempre con la misma estructura.
- Aplicar una metodología de auditoría.
- Seguir una guía interna de hardening.
- Redactar documentación técnica con un estilo determinado.
- Clasificar vulnerabilidades con criterios propios.
- Revisar código siguiendo una checklist.
- Trabajar con plantillas de cliente.
- Automatizar procedimientos repetibles.

La pregunta que debemos hacernos es: **¿Queremos que el agente haga esta tarea siempre de una forma concreta?** Si la respuesta es sí, probablemente necesitamos una Skill.

No son tecnologías competidoras

El error habitual es pensar que MCP server y Skill compiten entre sí. No compiten. Se complementan. Un MCP server nos da capacidad operativa. Una Skill nos da método. Si tenemos solo herramientas, el agente puede improvisar demasiado. Si tenemos solo metodología, el agente puede quedarse sin datos reales. Cuando combinamos ambas cosas, obtenemos agentes más útiles, repetibles y controlables. En términos prácticos:

```
Solo MCP: El agente puede actuar, pero puede no seguir nuestro método.
Solo Skill: El agente sabe cómo debería trabajar, pero puede no tener acceso a los sistemas.
MCP + Skill: El agente puede actuar y además seguir nuestro método.
```

El punto de seguridad

Hay que tener cuidado con ambas piezas.

- Un MCP server puede ser **peligroso si expone herramientas demasiado potentes**, si no limita permisos, si no valida entradas o si permite ejecutar acciones destructivas sin control.
- Una Skill también puede ser **peligrosa si incluye instrucciones maliciosas**, scripts inseguros o procedimientos que fuerzan al agente a tomar decisiones incorrectas.

Por eso, si vamos a usar MCP servers o Skills de terceros, deberíamos tratarlos como trataríamos cualquier dependencia externa: revisarlos, limitar permisos, ejecutarlos en entornos controlados y no darles acceso innecesario. En ciberseguridad esto es todavía más importante. Un agente con herramientas de red, acceso SSH, credenciales, filesystem y capacidad de generar comandos no es un chatbot inocente. Es un operador automatizado. Si lo conectamos mal, podemos crear un problema serio.

La idea importante

La forma más simple de recordarlo es esta:

```
MCP server = manos
Skill = manual
```

Si queremos que un agente de IA toque sistemas reales, necesitamos herramientas. Si queremos que trabaje con nuestro criterio, necesitamos metodología. Y si queremos que sea realmente útil en entornos técnicos, necesitamos las dos cosas bien separadas y bien integradas.