Ho dato le chiavi del server dei miei clienti a un'IA. Ecco come non è finita male

Di Fabio Angelici • Pubblicato il 16 luglio 2026 • 9 min di lettura

Automazione | AI nel Lavoro | Case Studies

RISPOSTA RAPIDA

Ho installato un agente AI (Claude Code) sul VPS dove ospito i siti di oltre 30 clienti, dandogli accesso reale al server. Invece di sudo pieno, gli ho creato un utente dedicato con permessi limitati a una lista precisa di script whitelistati, log di ogni azione e conferma obbligatoria per operazioni distruttive. I backup automatici restano su cron classico, non sull'AI. Risultato: meno lavoro ripetitivo per me, zero rischi inutili per i dati dei clienti.

Detta così suona irresponsabile. Ospito sul mio VPS i siti di oltre 30 clienti diversi,

e qualche settimana fa ho deciso di installarci sopra Claude Code, un agente AI capace di eseguire comandi reali sul server:

creare utenti, controllare la sicurezza, fare backup, aggiornare progetti.

Non un chatbot che suggerisce codice da incollare io.

Un agente che il codice lo esegue da solo, in autonomia, mentre io magari sto facendo altro.

Se il primo pensiero che ti è venuto è "e se sbaglia qualcosa?", hai ragione a pensarlo.

È esattamente la domanda che mi sono fatto anch'io prima di iniziare, e la risposta non è stata "gli do fiducia e vediamo cosa succede".

È stata costruire un'architettura che rende quasi impossibile un disastro, anche se l'agente si comportasse in modo imprevisto.

Questo articolo racconta come, e perché penso che sia un ragionamento utile anche per chi non ha mai sentito parlare di server o di riga di comando.

Perché l'ho fatto

Con più di 30 progetti attivi, buona parte delle mie giornate va in operazioni ripetitive:

controllare lo spazio disco, verificare le scadenze dei certificati SSL, capire se un cliente ha vulnerabilità note nelle dipendenze del suo sito,

fare provisioning di un nuovo progetto, pulire una cache dopo un deploy.

Compiti che non richiedono creatività, solo attenzione, il tipo di lavoro dove un errore umano da stanchezza

è più probabile di un errore della macchina, e dove ogni minuto speso è un minuto che non dedico a quello che davvero

fa la differenza per un cliente: capire cosa gli serve e costruirlo bene.

Volevo un assistente che potesse fare queste cose al posto mio, o almeno prepararmi il terreno prima che mi ci mettessi io.

Non volevo, però, un assistente con accesso illimitato al server dove vivono i dati e i siti di persone che si fidano di me.

Sono due desideri che sembrano in conflitto, e la maggior parte delle guide che trovi online su come installare un agente AI

su un server non ti aiuta granché a conciliarli:
o ti dicono "dagli pieno controllo, tanto è intelligente", oppure ti spaventano al punto da farti rinunciare del tutto.

L'errore che (quasi) tutti fanno con gli agenti AI

La scorciatoia più comune, quando installi un agente come Claude Code su un server, è dargli sudo completo,

cioè i permessi di amministratore su tutto il sistema, oppure, peggio ancora,

lanciarlo con le opzioni che saltano ogni conferma prima di eseguire un comando.

Anthropic stessa, nella documentazione ufficiale sulla sicurezza di Claude Code,

avverte chiaramente che questa modalità va usata solo in ambienti isolati, mai su un sistema con dati reali di clienti.

Funziona, nel senso che l'agente riesce a fare qualsiasi cosa.

Il problema è che "qualsiasi cosa" include anche cancellare un database per errore,

interpretare male un'istruzione ambigua,

o eseguire un comando suggerito da un contenuto esterno che l'agente ha letto senza che tu te ne accorgessi.

Non è un rischio teorico, ed è per questo che ho voluto raccontartelo con dati alla mano invece che con semplici allarmismi.

Negli ultimi mesi sono usciti diversi casi documentati da testate giornalistiche di agenti AI che,

lavorando in autonomia su compiti di sviluppo, hanno cancellato database di produzione interi,

in un caso riportato da Euronews, un agente ha impiegato nove secondi a distruggere sia i dati che i backup

di un'azienda che gestiva prenotazioni per decine di attività di noleggio auto,

semplicemente perché aveva trovato una credenziale con più permessi di quanto sarebbe dovuto avere,

e nessun sistema gli ha chiesto conferma prima di procedere.

In un altro caso, l'agente di sviluppo di una nota piattaforma ha cancellato il database di produzione di un utente

nonostante gli fosse stato esplicitamente vietato di toccare qualsiasi cosa durante un "blocco" temporaneo del sistema.

Il punto in comune tra questi episodi non è che l'AI fosse "cattiva" o incapace, i modelli usati erano tra i più avanzati disponibili.

Il punto è che nessuno aveva costruito una barriera reale tra quello che l'agente *poteva teoricamente fare*

e quello che *doveva davvero fare* per il compito assegnato.

L'organizzazione internazionale che si occupa di sicurezza delle applicazioni basate su AI,

OWASP, ha un nome preciso per questo: "eccessiva autonomia", cioè un agente a cui vengono dati più permessi,

più funzionalità o più libertà d'azione di quanti gliene servano davvero.

Più privilegi ha, più danno può fare un singolo errore, suo o tuo, nell'istruzione che gli hai dato.

Il principio che ho seguito invece

In sicurezza informatica esiste un principio vecchio di decenni, molto prima che esistessero gli agenti AI:

dare a ogni utente, processo o sistema solo il minimo di accesso necessario per fare il proprio lavoro, niente di più.

Il National Institute of Standards and Technology americano lo chiama "least privilege",

ed è uno dei capisaldi di ogni buona pratica di sicurezza informatica, vale per un dipendente con accesso ai sistemi aziendali,

vale per un account di servizio automatico, e vale altrettanto per un agente AI che esegue comandi su un server.

Concretamente, sul mio server questo si è tradotto in alcune scelte precise.

L'agente non usa mai il mio utente amministratore, ha un utente Linux tutto suo, separato e senza privilegi di default.

Non ha un accesso generico da amministratore: può eseguire solo una lista precisa di operazioni,

ognuna corrispondente a uno script che ho scritto e testato io in anticipo, creare un database, controllare i log di un progetto, fare un backup.

Non può inventarsi comandi nuovi, può solo lanciare quelli che ho già approvato uno per uno.

Ogni azione che compie viene registrata in un log, così so sempre esattamente cosa è successo e quando.

E per qualsiasi operazione che cancella o sovrascrive qualcosa, gli ho chiesto di fermarsi sempre e chiedermi conferma esplicita,

ogni singola volta, senza eccezioni.

Il risultato pratico è che, anche se gli chiedessi qualcosa di ambiguo, o se interpretasse male un'istruzione,

il danno massimo possibile resta contenuto dentro quello che i singoli script permettono.

Non può cancellare i dati di un cliente che non gli ho chiesto di toccare,

non può leggere le informazioni di un progetto diverso da quello su cui sta lavorando,

e non può nemmeno vedere il file dove è salvata la password del database, perché quella parte del sistema non è visibile al suo utente, punto e basta.

È la differenza tra dare a qualcuno le chiavi di tutta la casa e dargli solo quelle di una stanza specifica, con un compito preciso da svolgere lì dentro.

Le automazioni vere restano fuori dall'AI

Un punto su cui sono stato volutamente rigido: i backup automatici settimanali dei siti dei miei clienti non li fa l'agente AI.

Girano tramite un meccanismo tradizionale e ben collaudato, quello che nel gergo si chiama "cron",

presente su qualsiasi server da trent'anni, che esegue direttamente gli script di backup, senza alcun ragionamento nel mezzo.

L'AI è ottima per capire un problema nuovo, decidere come intervenire, gestire un'eccezione che non avevo previsto,

ma per un'operazione che deve succedere ogni domenica notte, sempre uguale, senza sorprese, preferisco uno script prevedibile

a un agente che "decide" cosa fare mentre nessuno lo sta guardando.

L'agente lo uso quando sono io a interrogarlo, in una sessione dove vedo ogni comando prima che parta e posso fermarlo in qualsiasi momento.

Per tutto il resto, la parte noiosa, ripetitiva, che deve succedere puntualmente anche quando dormo, preferisco un vecchio,

affidabile programma pianificato, che fa sempre e solo quello per cui è stato scritto.

Gli intoppi (perché la radical honesty vale anche qui)

Non è filato tutto liscio, ed è giusto dirlo, perché fingere che tutto sia andato perfetto al primo colpo sarebbe disonesto,

e non è lo spirito con cui scrivo su questo blog.

Durante i test, l'agente stesso ha scoperto che due dei miei script partivano da un presupposto sbagliato:

che il nome della cartella di un progetto coincidesse sempre con lo username Linux associato.

Un'assunzione vera per i progetti che ho creato con le procedure automatiche più recenti, falsa per cinque di quelli vecchi,

creati a mano anni fa con convenzioni di naming diverse.

Se non me ne fossi accorto, avrei rischiato di eseguire operazioni sull'utente sbagliato senza saperlo.

Ho scoperto anche un bug molto più sottile in uno script di backup:

segnalava un fallimento anche quando il backup era andato perfettamente a buon fine,

per un dettaglio tecnico di come la shell del sistema gestisce l'esito dell'ultima riga di uno script.

Sembra una sciocchezza, ma è esattamente il tipo di falso allarme che, ripetuto abbastanza volte, finisce per farti ignorare anche gli allarmi veri.

Nessuno di questi problemi ha causato danni reali, proprio perché l'architettura a permessi limitati ha reso ogni errore visibile e contenuto,

invece che silenzioso e distruttivo.

Se stai pensando di dare accesso a un agente AI a un sistema che gestisce dati di clienti veri, questa è la parte che conta davvero:

non "l'AI è abbastanza intelligente da non sbagliare mai", perché prima o poi sbaglierà, come sbagliamo tutti.

Conta che il sistema sia costruito in modo che, quando l'errore arriva, non succeda niente di grave, e che tu te ne accorga subito, non tre mesi dopo.

Se vuoi provarci anche tu

Qualche punto fermo, prima di installare qualsiasi agente AI su un server di produzione, che sia il tuo o quello di un'agenzia per cui lavori:

- Mai l'utente root, mai il tuo utente amministratore personale: crea sempre un utente dedicato solo all'agente
- Mai un accesso amministrativo generico: whitelist di comandi specifici, uno script per compito, niente di più
- Mai l'opzione che salta le conferme prima di eseguire un comando, su un ambiente con dati reali di persone vere
- Sempre un piano B per accedere al server se qualcosa va storto, una console d'emergenza del tuo provider,
- una chiave di accesso di riserva conservata al sicuro
- Le automazioni davvero critiche, quelle che devono succedere sempre e comunque, restano su un meccanismo tradizionale e prevedibile,
- non nel ciclo decisionale di un agente autonomo

Non è la strada più veloce, e all'inizio richiede più lavoro di configurazione rispetto a "installa e dagli accesso a tutto".

Ma è quella che ti fa dormire la notte sapendo che i dati dei tuoi clienti sono al sicuro anche il giorno in cui qualcosa va storto perché,

che si tratti di un'intelligenza artificiale o di un essere umano stanco alle due di notte, prima o poi qualcosa va sempre storto.

La differenza la fa quanto lontano può arrivare quell'errore prima che qualcuno (tu, o il sistema stesso) riesca a fermarlo.

FONTI

1. Documentazione ufficiale sulla sicurezza di Claude Code
2. Un agente in 9 secondi distrugge dati e backup - Euronews
3. Agente cancella database di produzione di un utente - Fortune
4. OWASP
5. National Institute of Standards and Technology
