Le due facce dello stesso rischio CI/CD: Cordyceps e la falla agentica di Claude Code



La sicurezza CI/CD ha due facce, e due ricerche pubblicate a poche settimane di distanza le raccontano da angolazioni opposte come un'unica storia. C'è *Cordyceps*, la classe di vulnerabilità nei *workflow* CI/CD che colpisce oltre 300 repository GitHub pienamente sfruttabili, con riscontri confermati su progetti di Microsoft, Google, Apache e Cloudflare. E c'è la falla individuata nella GitHub Action di Claude Code, in cui un agente di intelligenza artificiale viene convinto a leggere e a far uscire i segreti del *runner*. Tecnicamente sono fenomeni diversi: una composizione insicura di automazioni deterministiche, l'altra un *prompt injection* contro un agente. La radice, però, è identica, e capirla conta più di entrambi i singoli casi.

La radice comune della sicurezza CI/CD: il workflow è codice

Il punto di partenza è una svista concettuale diffusa: trattare i file di automazione (gli *YAML* di GitHub Actions e i loro equivalenti) come semplice configurazione, e non come codice critico per la sicurezza. Questi *workflow* eseguono comandi di shell, custodiscono chiavi di firma, si autenticano verso i cloud provider e pubblicano release. In mezzo c'è sempre un confine di fiducia: input che arriva dall'esterno (una *pull request*, un commento, il corpo di una *issue*) e automazioni che girano con permessi elevati e accesso ai segreti. Quando quel confine non e' presidiato, l'input non attendibile attraversa la barriera e raggiunge ciò che dovrebbe restargli precluso. *Cordyceps* e la falla di Claude

Code sono due modi diversi di far attraversare quel confine alla stessa cosa sbagliata.

Prima faccia: Cordyceps, la composizione insicura

Nella ricerca di Novee pubblicata il 23 giugno il problema non è un bug puntuale con un CVE, è un pattern sistemico: configurazioni che concedono alle *pull request* più permessi del dovuto, così che una richiesta non attendibile attiva *workflow* privilegiati. Le catene più pericolose sono multi-step: un *workflow* a basso privilegio passa il proprio output a uno ad alto privilegio, il cui *token* si autentica al cloud con il ruolo più elevato (
roles/owner
su GCP, in uno dei casi confermati). Nessun passaggio, isolato, sembra pericoloso; la vulnerabilità vive solo nella composizione. Su circa 30.000 repository scansionati, Novee ne ha segnalati 654 in una singola passata e ne ha confermati oltre 300 pienamente sfruttabili, con un dettaglio dirimente: si tratta di sfruttabilità provata con *proof of concept* in laboratorio, non di abusi osservati in rete. E’ la stessa logica che ICT Security Magazine ha già documentato con l’attacco a LiteLLM, dove uno scanner di sicurezza compromesso in GitHub Actions è diventato il punto di ingresso alla *supply chain*.

Un tratto rende *Cordyceps* particolarmente insidioso: l’exploit non richiede un account privilegiato. Secondo Novee basta un utente anonimo con un profilo gratuito per forgiare approvazioni, iniettare codice o sottrarre credenziali.

Seconda faccia: l’agente che legge

troppo

Il secondo caso aggiunge un attore nuovo dentro la pipeline: un agente AI che interpreta testo. La GitHub Action di Claude Code esegue Claude dentro il *runner* di CI per smistare *issue*, applicare etichette e revisionare *pull request*, e per impostazione predefinita dispone di accesso in lettura e scrittura a codice, *issue*, *pull request* e file di *workflow*. La ricerca di GMO Flatt firmata da RyotaK ha mostrato come una singola *issue* malevola, sfruttando un *prompt injection* indiretto, possa indurre l'agente a leggere /proc/self/environ e a riscrivere i valori nella *issue* stessa, dove l'attaccante li raccoglie. Il bottino più prezioso non e' la chiave API in sé: sono le credenziali con cui il *workflow* richiede il *token* OIDC, che Claude Code scambia con il backend di Anthropic per un *token* di installazione della GitHub App con permessi di scrittura, fino al controllo del repository. Poichè lo stesso *workflow* girava anche sul repository ufficiale dell'azione, un attacco riuscito avrebbe potuto iniettare codice nell'azione stessa, con effetto a cascata su tutti i progetti a valle.

La analisi di Microsoft Threat Intelligence, pubblicata il 5 giugno, ha messo a fuoco il dettaglio architetturale: lo strumento *Read* non era soggetto allo stesso *sandboxing* applicato all'esecuzione via *Bash*, e poteva quindi leggere /proc/self/environ
e ricavarne ANTHROPIC_API_KEY
. Per aggirare i filtri di sicurezza del modello e il *secret scanning* di GitHub, l'iniezione mascherava la richiesta come revisione di conformità e ordinava di tagliare i primi sette caratteri della chiave, neutralizzando il prefisso sk-ant-
che ne avrebbe tradito la natura. Anthropic ha corretto la falla principale segnalata da RyotaK in quattro giorni, con ulteriore *hardening* in claude-code-action v1.0.94
, e ha mitigato la variante di Microsoft in Claude Code 2.1.128
il 5 maggio, bloccando l'accesso ai file sensibili in /proc

; le vulnerabilità sono state valutate 7.8 in CVSS v4.0 e premiate con un *bug bounty*. Lo scenario non è solo teorico: come riportato da The Hacker News, a febbraio un titolo di *issue* avvelenato nel *workflow* di triage di Cline aveva permesso il furto di un *token* di pubblicazione npm e la diffusione di una versione non autorizzata, ritirata dopo circa otto ore.

Perchè è lo stesso problema

In entrambi i casi un input esterno non attendibile attraversa un confine di fiducia ed entra in un'automazione privilegiata che custodisce segreti. *Cordyceps* lo fa per via deterministica, incatenando *workflow* fino a un *token* ad alto privilegio. Il caso agentico ottiene lo stesso esito perchè un agente AI interpreta il testo dell'attaccante come istruzione: non serve nemmeno una catena mal configurata, basta convincere l'agente a leggere il segreto. E' lo stesso punto cieco che la rivista ha raccontato a proposito delle skill AI malevole, dove un componente caricato da un agente viene eseguito con la sua autorità. C'è anche un dettaglio rivelatore comune: nessuno dei due ha ricevuto un CVE, perchè si tratta di rischi di architettura e di composizione, non di difetti puntuali. E' la categoria di problema che gli strumenti tradizionali, abituati ad analizzare un file alla volta, faticano a vedere.

Le difese valgono per entrambe le facce

La buona notizia è che le contromisure si sovrappongono quasi del tutto. Trattare i *workflow* come codice e non come configurazione; applicare il privilegio minimo a *token* e ruoli; isolare i confini di fiducia fra automazioni a basso e ad alto privilegio; non eseguire mai *workflow* automatici su *pull request*, *issue* o commenti esterni senza un controllo di accesso; validare e sanificare gli input controllabili

dall’attaccante. Per la componente agentica, Microsoft affianca la *Agents Rule of Two* (originata da un framework di Meta del 31 ottobre 2025): un *workflow* AI non dovrebbe mai detenere contemporaneamente le tre capacità di elaborare input non attendibile, accedere a dati sensibili e modificare stato o comunicare verso l’esterno. Il principio operativo, sintetizzato dalla documentazione di hardening di GitHub Actions, resta lo stesso: ogni input va trattato come ostile e ogni permesso va ridotto al minimo indispensabile.

Una traiettoria, non due incidenti

Per chi guida la sicurezza la lezione non è scegliere quale delle due falle temere di più. E’ riconoscere che la *supply chain* del software inizia oggi nei file CI/CD, e che la diffusione della programmazione assistita da agenti moltiplica entrambe le facce: schemi insicuri replicati a velocità industriale, e agenti che leggono testo non attendibile con permessi elevati. Con il Cyber Resilience Act e la NIS2 che spostano la sicurezza della catena di fornitura dal piano delle buone pratiche a quello degli obblighi, presidiare il confine di fiducia dei *workflow* smette di essere un tema da addetti ai lavori e diventa una questione di conformità

Condividi sui Social Network:
https://www.ictsecuritymagazine.com/notizie/due-facce-rischio-cicd-cordyceps-claude-code/