IaC security: la falla nel cloud nasce nel codice che lo costruisce

IaC security è la disciplina che sposta la sicurezza del cloud dove il cloud viene davvero deciso: nel codice che lo descrive. L'*infrastructure as code*, la pratica di definire server, reti, permessi e archivi in file versionati invece che a mano da una console, ha reso ripetibile e scalabile la costruzione delle infrastrutture. Ma ha anche spostato il punto in cui nascono gli errori. Una configurazione sbagliata, oggi, non è quasi mai una svista compiuta su una macchina in produzione: è una riga in un file di *infrastructure as code* che provvederà a replicarla, identica, su ogni risorsa che genera.

Il dato che inquadra la posta è noto e ruvido. Una previsione di Gartner, formulata per il quinquennio chiusosi nel 2025, lo fissava in termini netti: la quasi totalità dei fallimenti di sicurezza nel cloud sarebbe stata responsabilità del cliente, non del fornitore, con le configurazioni errate come causa prevalente. La finestra di quella previsione è ormai conclusa, ma la sua sostanza è confermata dalle rilevazioni più recenti, che continuano a indicare nella misconfigurazione la prima causa delle violazioni cloud. Se questo è vero, allora il luogo più economico e più efficace in cui intervenire non è la nuvola già costruita, ma il progetto che la costruisce. La IaC security parte esattamente da qui: correggere la falla quando è ancora una riga di codice, prima che diventi una risorsa esposta.

La configurazione errata nasce due passi prima

Per anni la sicurezza del cloud ha guardato al risultato. Gli strumenti di gestione della postura ispezionano l'ambiente in esecuzione alla ricerca di archivi pubblici, permessi troppo larghi, database non cifrati, e segnalano ciò che trovano. È utile, ma è una diagnosi a valle: la risorsa pericolosa esiste già, è stata creata, magari è online da ore o da giorni. La domanda che la IaC security pone è diversa: perché aspettare che la configurazione errata esista, se è scritta nero su bianco nel codice che la genererà?

Spostare il controllo a sinistra, verso il momento in cui l'infrastruttura viene scritta anziché eseguita, cambia l'economia della correzione. Un errore intercettato nel codice si risolve modificando una riga, prima che tocchi un solo sistema reale; lo stesso errore scoperto in produzione richiede di rimediare a una risorsa viva, capire chi vi ha già avuto accesso, gestire l'incidente. È la stessa logica del secure coding applicata non al software, ma all'infrastruttura che lo ospita: il difetto costa meno dove nasce.

Scansionare l'infrastruttura prima che esista

In pratica, la IaC security comincia dalla scansione statica del codice infrastrutturale. Strumenti come Checkov, tfsec e il suo successore Trivy analizzano i file di Terraform, CloudFormation o i manifesti di Kubernetes e li confrontano con ampie librerie di regole, comprese quelle derivate dai benchmark di conformità, segnalando le configurazioni pericolose prima del rilascio. Un archivio reso pubblico, una cifratura mancante, un gruppo di sicurezza aperto al mondo intero: tutto questo compare nel codice, e lì può essere bloccato.

Il valore sta nell'integrazione con il flusso di lavoro dello sviluppo. Eseguita come controllo prima del *commit*, la scansione restituisce allo sviluppatore un riscontro immediato, mentre sta ancora scrivendo, quando correggere è banale. Ripetuta nella pipeline di integrazione continua, diventa un cancello che impedisce a una modifica non conforme di essere unita al ramo principale. È il cuore di un approccio DevSecOps maturo: la sicurezza non come revisione finale che rallenta, ma come riscontro continuo che si muove alla velocità del codice. La configurazione errata, in questo schema, non viene rimediata in produzione perché in produzione non ci arriva.

IaC security e policy as code: dalle scansioni alle regole proprie

La scansione coglie ciò che è notoriamente sbagliato, ma ogni organizzazione ha regole proprie che nessuna libreria generica conosce: nessun archivio pubblico senza eccezione approvata, cifratura obbligatoria ovunque, etichette di proprietà su ogni risorsa, reti che non possono comunicare. Esprimere e imporre queste regole è il compito del *policy as code*, il livello in cui la IaC security smette di reagire a difetti noti e inizia a governare attivamente ciò che è permesso costruire.

Lo strumento più trasversale è Open Policy Agent, progetto graduato della CNCF, che con il linguaggio dichiarativo Rego consente di scrivere le politiche come codice, versionarle, testarle e applicarle in modo uniforme; in ambito Kubernetes gli si affiancano motori nativi come Kyverno. Le regole, descritte nella documentazione di OPA, si applicano dove servono: nella pipeline, per validare un piano Terraform prima che venga eseguito, o al momento dell'ammissione di una risorsa. Come spiega la stessa introduzione CNCF al policy as code, il vantaggio è la coerenza: una politica scritta una volta vale per ogni progetto, ogni team, ogni ambiente, e non

dipende più dalla diligenza del singolo. Sono guardrail, barriere che incanalano, non cancelli arbitrari che bloccano: lo sviluppatore resta libero di muoversi, purché dentro i confini che l'organizzazione ha deciso.

Il problema del drift

C'è un limite che la IaC security onesta non nasconde. Il codice descrive lo stato desiderato dell'infrastruttura, ma la realtà in esecuzione tende a discostarsene: una modifica fatta a mano nella console per risolvere un'urgenza, un aggiustamento mai riportato nel codice, e l'infrastruttura reale comincia a divergere da ciò che il codice dichiara. È il fenomeno del *drift*, lo scostamento, e dove esiste rende il codice una finzione: si può aver messo in sicurezza ogni riga e avere comunque, in produzione, una risorsa configurata diversamente da come il codice crede. Per questo la disciplina si completa con il rilevamento del *drift*, il confronto periodico tra stato dichiarato e stato effettivo, che riconcilia i due o segnala lo scarto. Mettere in sicurezza il codice serve a poco se nessuno verifica che la realtà gli somigli ancora.

La supply chain dell'infrastruttura come codice

Il secondo fronte aperto è quello dei moduli. Scrivere infrastruttura da zero è raro: si riusano moduli pubblici e componenti di terze parti, esattamente come si fa con le librerie del software. Ma un modulo di *infrastructure as code* è una superficie d'attacco particolare, perché viene eseguito con i privilegi dell'infrastruttura: un modulo Terraform malevolo o compromesso può creare risorse in un account cloud, modificare le politiche di gestione degli accessi, aprire percorsi di rete, il tutto con i permessi più alti. Importare un modulo non verificato significa importare codice con

diritti da amministratore, e vale per l’IaC la stessa attenzione alla catena di fornitura che ormai si dedica al software applicativo: tracciare le dipendenze, fissarne le versioni, valutarne la provenienza.

Messa insieme, la IaC security ridefinisce il baricentro della sicurezza del cloud. Sposta l’attenzione dall’ispezione del risultato al governo della sorgente, perché è nella sorgente che la configurazione errata, prima causa delle violazioni cloud, prende forma. Scansionare il codice, esprimere le regole come *policy as code*, sorvegliare il *drift* e curare la catena dei moduli sono le quattro facce di un unico spostamento: smettere di rincorrere le falle nell’infrastruttura viva e impedirle nel codice che la genera. Il cloud, in fondo, è fatto di codice, e mettere in sicurezza il cloud vuol dire, prima di tutto, mettere in sicurezza quel codice.

Condividi sui Social Network:
https://www.ictsecuritymagazine.com/cyber-security/iac-security-policy-as-code/