Confidential Computing su AWS: Proteggere i dati in esecuzione con Nitro Enclaves

7 Aprile 2026 - 9 min. read

AWS Nitro Enclaves | Confidential Computing | Data Protection

La sicurezza in Cloud ha raggiunto una certa maturità, garantendo un'elevata affidabilità e consentendo di rispettare gran parte delle checklist di sicurezza con relativamente poco effort. Negli anni sono nati strumenti e best practices; si è imparato a mettere in sicurezza il perimetro della rete virtuale, a gestire le identità in modo granulare e a fare ampio e diffuso uso della crittografia forte per cifrare i dati in transito e at rest.

Eppure, per molto tempo, è rimasto un punto cieco critico in questa fortezza digitale, un momento di inevitabile vulnerabilità in cui i dati, per essere effettivamente elaborati dai processori, devono essere in memoria non cifrati.In questo articolo si esplorerà il concetto di **Confidential Computing**, il paradigma tecnologico che ha definitivamente chiuso questo gap di sicurezza, ridefinendo gli standard di protezione nel Cloud. Si vedrà qual è la complessa sfida tecnologica alla base della protezione dei dati durante l'elaborazione, perché i tradizionali container - su cui poggiano quasi tutte le architetture moderne – non sono sufficienti per i workload ultra- sensibili, e come **AWS Nitro Enclaves** offra una soluzione elegante e inattaccabile per creare vere e proprie "casseforti" crittografiche per le applicazioni più critiche.

Oltre lo storage e la rete: cos'è il Confidential Computing?

Per comprendere appieno l'importanza e l'impatto del Confidential Computing, è necessario guardare l'intero ciclo di vita del dato attraverso la lente del

cosiddetto **Trilemma della Protezione Dati**. Fino a poco tempo fa, gli strumenti a disposizione potevano risolvere con successo solo due di questi tre punti:

1. **Data at Rest (Dati a riposo):** Quando i dati sono archiviati su un database relazionale, su un volume a blocchi (Amazon EBS) o su uno storage a oggetti (come Amazon S3), vengono protetti tramite robusti algoritmi di cifratura simmetrica (es. AES-256) integrati con servizi di Key Management come AWS KMS. Se qualcuno rubasse fisicamente un disco rigido nel data center, o se un bucket venisse configurato in modo errato, l'attaccante si ritroverebbe di fronte a un mero ammasso di bit incomprensibili e inutilizzabili.
2. **Data in Transit (Dati in transito):** Quando i dati viaggiano in rete, dal dispositivo di un client al back-end o lateralmente tra i microservizi all'interno di una VPC, si utilizzano protocolli crittografici consolidati come TLS/SSL (spesso in modalità *mutual TLS*). Questo tunnel cifrato evita che i dati vengano intercettati, spiati o manipolati durante il tragitto, ad esempio tramite attacchi del tipo *man-in-the-middle* o *sniffing*.
3. **Data in Use (Dati in uso):** Qui arriva la vera sfida, l'anello debole della catena. Affinché le applicazioni possano processare i dati, che si tratti di effettuare una ricerca all'interno di un database e formattare i dati per inviarli al FE, calcolare il totale di una transazione finanziaria, o eseguire l'inferenza di un modello di Intelligenza Artificiale su referti medici — **i dati devono risiedere non cifrati nella memoria RAM** e nei registri del processore. In quel preciso istante, il dato è vulnerabile.

Il **Confidential Computing** è l'insieme di tecnologie, sia hardware che software, progettate appositamente per proteggere questi *Data in Use*. Utilizzando un isolamento profondo basato su componenti hardware specializzati, viene creato un **TEE (Trusted Execution Environment)**. Un TEE è un ambiente di esecuzione sicuro, isolato a livello di processore, in cui i dati in chiaro e il codice che li elabora non possono essere né visti, né esfiltrati, né alterati in alcun modo dall'esterno. Nemmeno da chi detiene i privilegi massimi di amministrazione sull'infrastruttura sottostante.

Oggi, in un'era dominata dall'addestramento di modelli LLM proprietari e dalla *Multi-Party Computation* (dove diverse aziende collaborano unendo i propri dataset sensibili per trarne insight, senza volerli però mostrare in chiaro ai partner), il Confidential Computing non è più una tecnologia di nicchia per enti governativi o bancari.

L’isolamento dei container NON protegge i dati in use

Facendo un passo indietro e analizzando esattamente la dinamica del problema che il Confidential Computing si prefigge di risolvere, in un'architettura Cloud basata su macchine virtuali (istanze EC2) che fanno girare flotte di Container orchestrati da piattaforme come Docker o Kubernetes, è necessario porsi una domanda scomoda: chi oltre a me può accedere al container?

La risposta è l'amministratore di sistema dell'istanza ospitante, o per meglio dire, l'utente **Root**.

È fondamentale interiorizzare che l'isolamento offerto dai Container è di tipo puramente logico (basato su *namespaces* e *cgroups*), non fisico. Più container che girano sulla stessa istanza EC2 condividono lo stesso Kernel e, in fin dei conti, la stessa RAM (sebbene sezionata a livello logico). Questo significa che se un malintenzionato - o anche solo un *insider threat* come un dipendente corrotto, o un malware che ha compiuto una *privilege escalation* - riesce ad ottenere i privilegi di Root sull'istanza che ospita i container, la partita della sicurezza è persa.

Ma i container non sono il villain di questa storia, infatti, evitarli e ripiegare su processi eseguiti direttamente su un'istanza EC2 tradizionale non risolve il problema alla radice; anzi, espone a vettori d'attacco ancor più diretti. In questo scenario, infatti, non è nemmeno necessario che l'attaccante ottenga privilegi di Root a livello di sistema operativo. È sufficiente compromettere il singolo utente di sistema con cui l'applicazione è in esecuzione: sfruttando una falla di sicurezza nel codice (ci sono diversi tipi di attacco possibili), l'attaccante può ottenere l'accesso legittimo alla porzione di memoria RAM assegnata a quel processo.

In entrambi i casi, una volta ottenuto l'accesso alla memoria, l'esito è identico; è possibile eseguire un **dump completo della memoria RAM** associata al processo bersaglio. In una frazione di secondo, si può salvare su disco un file contenente esattamente tutto ciò che un'applicazione stava elaborando in chiaro in quel momento. Analizzando poi il dump, l'attaccante può estrarre svariate informazioni critiche e, magari, sensibili: token di sessione JWT attivi, password degli utenti, *seed* crittografici per la generazione di codici TOTP, chiavi private SSL e informazioni di identificazione personale (PII).

Il concetto chiave qui è il **TCB** (*Trusted Computing Base*). Il TCB rappresenta l'insieme di tutte le entità hardware, software e umane di cui ci si deve "fidare" ciecamente per garantire la sicurezza del sistema. In uno scenario basato su container tradizionali, il TCB include l'infrastruttura del Cloud Provider, il livello di Hypervisor, l'intero sistema operativo Host, i demoni di orchestrazione (kubelet, dockerd) e tutti gli amministratori di sistema. Per workload realmente mission-critical e sensibili, questa platea potrebbe risultare eccessivamente ampia.

La soluzione: AWS Nitro Enclaves

Per abbattere drasticamente questo **TCB,** entra in gioco **AWS Nitro Enclaves**. Questo servizio poggia sulle fondamenta della tecnologia AWS Nitro System, l'architettura hardware personalizzata di AWS che ha rivoluzionato il cloud computing disaccoppiando fisicamente le funzioni di rete, storage e gestione dalla CPU principale dell'istanza. Nitro Enclaves sfrutta questa architettura per permettere di "ritagliare" e creare partizioni totalmente isolate di CPU e Memoria direttamente da un'istanza EC2 "Parent" esistente.

Un'Enclave non va confusa con un container "più sicuro" o una VM alleggerita; è un paradigma architetturale e concettuale completamente diverso:

- **Isolamento a livello di silicio:** CPU e RAM assegnate in fase di avvio all'Enclave vengono letteralmente e fisicamente rimosse dalla disponibilità e dalla mappatura dell'istanza Parent. Il sistema operativo host (e di conseguenza l'onnipotente utente Root) semplicemente "non vede" più quella porzione di memoria, come se non fosse mai stata installata sulla scheda madre. Eseguire un dump di una memoria inesistente per l'host è tecnicamente impossibile.
- **Nessun accesso interattivo:** All'interno dell'Enclave non viene eseguito alcun server SSH o demone di accesso remoto. Non c'è una shell a cui connettersi, non c'è una console di emergenza. Nessuno, nemmeno lo sviluppatore capo che ha scritto l'applicazione, l'amministratore dell'account AWS o il supporto tecnico di AWS stesso, ha il potere di "entrarci" per guardare cosa succede. Questa è la vera incarnazione del principio di *Zero Access*.
- **Nessuna rete esterna:** L'Enclave, per design, non possiede alcuno stack di rete TCP/IP, né un indirizzo IP, né instradamento verso la VPC o Internet. L'unico modo in cui il codice confidenziale può comunicare con il mondo esterno (ovvero l'istanza Parent) è tramite un canale di comunicazione bidirezionale a livello di hypervisor chiamato **Vsock**. L'istanza Parent dovrà quindi agire come un proxy rigorosamente

controllato per inoltrare input e output, impedendo qualsiasi attacco di rete diretto all'Enclave.

- **Attestazione crittografica (Attestation):** Prima di avviare l'elaborazione, l'Hypervisor Nitro calcola in modo deterministico una serie di hash (PCR) che "provano" matematicamente l'identità e l'integrità del codice in esecuzione e del kernel dell'Enclave. L'Enclave può usare questa "carta d'identità crittografica" per ottenere da AWS KMS chiavi di decifratura o altri segreti. I PCR possono essere inclusi nelle policy di autorizzazione, garantendo che nessuno abbia iniettato codice malevolo o alterato l'immagine originale.

Se l'istanza EC2 Parent venisse totalmente compromessa da un hacker, l'attaccante troverebbe al massimo il software proxy che fa da ponte verso l'Enclave, ma il "motore" che macina i dati sensibili, insieme alle chiavi in chiaro e ai dati non cifrati, rimarrebbe custodito in una scatola nera impenetrabile.

All'atto pratico: usare Nitro Enclaves

Nonostante l'enorme complessità crittografica e l'isolamento hardware sottostante, AWS ha lavorato per rendere la *Developer Experience* sorprendentemente semplice e accessibile grazie allo strumento a riga di comando nitro-cli. Il grande vantaggio per i team di sviluppo è che non è necessario imparare nuovi linguaggi o riscrivere le applicazioni da zero: il punto di partenza è sempre un'immagine Docker standard.

Di seguito i tre passaggi essenziali per pacchettizzare e avviare un processo confidenziale.

1. Preparazione dell'immagine

Supponendo di avere un'applicazione containerizzata chiamata my-secure-app che contiene la logica per decifrare ed elaborare dati sensibili, il primo passo è convertire la classica immagine Docker in un formato speciale e verificabile, chiamato EIF (*Enclave Image Format*). Direttamente sull'istanza EC2 (precedentemente configurata tramite Launch Template per supportare le Enclaves), si lancia il comando di build:

```
nitro-cli build-enclave \
    --docker-uri my-secure-app:latest \
    --output-file my-secure-app.eif
```

Questo comando impacchetta l'applicazione insieme a un kernel Linux minimale, genera il file .eif risultante e, cosa fondamentale per la sicurezza, restituirà a schermo le misurazioni dei **PCR**. Questi hash crittografici identificano in modo univoco l'ambiente appena creato. Questi specifici hash saranno quelli da configurare minuziosamente nelle *Condition Keys* delle policy di AWS KMS per sbloccare l'Attestazione.

2. Esecuzione dell'Enclave

Una volta ottenuta l'immagine EIF, si può avviare l'Enclave. Questo comando andrà fisicamente a sottrarre i core della CPU e i megabyte di RAM dall'istanza Parent, assegnandoli esclusivamente al nuovo ambiente blindato:

```
nitro-cli run-enclave \
    --eif-path my-secure-app.eif \
    --cpu-count 2 \
    --memory 1024 \
    --enclave-cid 16
```

Il parametro fondamentale qui è --enclave-cid, che assegna un identificativo numerico (Context ID) all'Enclave. Si utilizzerà questo specifico CID all'interno dell'istanza Parent per instaurare la comunicazione con l'Enclave attraverso il Vsock.

3. Verifica dello stato

Per controllare che l'Enclave sia regolarmente in esecuzione (ricordando che l'ambiente è isolato e non si possono usare i classici comandi di orchestrazione come docker ps o monitoraggi tradizionali), è possibile interrogare l'allocatore Nitro lanciando:

```
nitro-cli describe-enclaves
```

Questo comando restituirà lo stato dell'Enclave, l'uptime e le risorse allocate. Se a questo punto si provasse, simulando il comportamento di un attaccante, a spulciare nei processi attivi del sistema operativo dell'istanza Parent tramite ps aux, o a forzare un dump della RAM, non si troverebbe alcuna traccia del binario in esecuzione, né tanto meno dei dati in chiaro gestiti all'interno di my-secure-app.eif.

Conclusioni

Il passaggio al Confidential Computing rappresenta un reale e profondo rafforzamento del paradigma di "Trust" tra i cloud provider, le aziende che sviluppano software e gli utenti finali che affidano loro le proprie informazioni più intime.

Rimuovere definitivamente l'amministratore di sistema e l'infrastruttura sottostante dall'equazione della fiducia, riuscendo a blindare i dati anche e soprattutto mentre vengono elaborati, è l'unica chiave per sbloccare nuovi casi d'uso rivoluzionari. Si pensi alla condivisione sicura di proprietà intellettuale per l'addestramento collaborativo di modelli di Intelligenza Artificiale, o all'analisi incrociata di dati bancari condivisi tra istituti per la prevenzione delle frodi.

Con AWS Nitro Enclaves, costruire questi inespugnabili "caveau" digitali non è più un esperimento di laboratorio, ma è diventato parte integrante e fluida del normale ciclo di sviluppo software.

About Proud2beCloud

Proud2beCloud è il blog di beSharp, APN Premier Consulting Partner italiano esperto nella progettazione, implementazione e gestione di infrastrutture Cloud complesse e servizi AWS avanzati. Prima di essere scrittori, siamo Solutions Architect che, dal 2007, lavorano quotidianamente con i servizi AWS. Siamo innovatori alla costante ricerca della soluzione più all'avanguardia per noi e per i nostri clienti. Su Proud2beCloud condividiamo regolarmente i nostri migliori spunti con chi come noi, per lavoro o per passione, lavora con il Cloud di AWS. Partecipa alla discussione!

Alessio Gandini

Cloud-native Development Line Manager @ beSharp, DevOps Engineer e AWS expert.Computer geek da quando avevo 6 anni, appassionato di informatica ed elettronica a tutto tondo. Ultimamente sto esplorando l'esperienza utente vocale e il mondo dell'IoT.Appassionato di cinema e grande consumatore di serie TV, videogiocatore della domenica.
