Sicurezza nell'era dei container

La sicurezza nell'era della Digital Trasformation e devOps subisce forti cambiamenti e "potenziali" semplificazioni evolvendo la logica di basic-virtualization verso una stagione di virtualizzazione applicativa basata sul modello a container. Ma come sempre, dietro all'evoluzione tecnologica si nascondono nuove minacce che cercheremo in questo articolo di "intravedere".

L'utilizzo di strumenti quali Docker, Rkt, *, che non sono altro che una semplificazione ed evoluzione del concetto di LXC già integrato in ambiente linux (primitive cgroup, scritte dagli ingegneri Google nel 2006 e rilasciati dal kernel 2.6.24 unite ai Namespace che implementano i concetti di isolamento e a chroot) garantiscono un aumento dell'isolamento delle componenti applicative riducendo in maniera drastica la presenza di Hypervisor e – teoricamente – anche della superfice di esposizione al rischio delle piattaforme.

Al netto delle problematiche di questa nuova tecnologia che risulta ad oggi ancora in evoluzione in termini di flessibilità e dvOps su architetture enterprise (soprattutto in ambito di operation), analizziamo i vantaggi promessi dalla filosofia a "container":

- Minori risorse occupate rispetto alla Basic-Virtualization
- Disaccoppiamento applicativo, versioning e portabilità.
- Vantaggi in operation (ad es. orchestration, riavvio, spegnimento, provisioning, cloning, routing, *)
- Completamento dell'implementazione del concetto di PaaS.

Ma esistono anche dei vantaggi tangibili in termini di sicurezza?

Dipende dalle scelte architetturali e applicative che verranno fatte, dalla corretta gestione dello sviluppo sicuro, dalla scelta del cloud da utilizzare e degli hypervisor di mediation, ma senza addentrarci troppo analizziamo la logica “basic” dell’approccio a container. Essendo ogni container tecnicamente isolato, non avendo accesso ai processi degli altri container e al sistema Host che li gestisce, cominciamo ad intravedere dei benefici in termini di sicurezza:

1. **Patching & Hardening:** in assenza di hypervisor, sicuramente si riduce il numero di Host e Guest da gestire. In una logica on-premise a container ridurremo di molto il numero di macchine virtuali e probabilmente riusciremo ad implementare finalmente una corretta gestione degli interventi di patching riducendo i volumi ma aumentando la qualità percepita. Ovviamente al netto dei benefici in termini di sicurezza, dovrà risultare tangibile la riduzione dei costi sull’hardware, licenze, energia e opex in generale.
2. **Sviluppo sicuro del Codice:** essendo i container isolati a livello applicativo vulnerabilità quali XSS, Injection, Path Traversal, * saranno pressoché difficili da ottenere per aver accesso alla macchina Host. Ad esempio i famosi path Traversal per l’accesso in lettura del password o dello shadow dei server saranno degli exploit forse da ricordare?
3. **Database:** i database noSQL in logica a micro-servizio (qualora dalle analisi applicative possibile) devono essere utilizzati e possono essere un ottimo spunto per migliorare la sicurezza in quanto i container avranno al loro interno sia la componente applicativa (ad esempio apache-tomcat) che la componente dati (ad esempio mongo db, Redis, Cassandra, *) per la sola quota parte in gestione del micro-servizio stesso. In questa ottica il micro-servizio accederà solo ai dati presenti nel DB

noSQL all’interno del proprio container pertanto eventuali violazioni di accesso non potranno violare l’intera base dati utilizzata e condivisa da tutti gli altri container.

E ovviamente, quando ci sono dei benefici devono essere previsti dei livelli di attenzione su specifici temi, di seguito alcuni esempi:

1. **Kernel exploit & Kernel Panic:** Attenzione agli attacchi su base Kernel. Circa il 45% delle vulnerabilità sul kernel Linux sono di DOS. Violando da un container il kernel (o anche violando le primitive di isolamento LXC), visto che tale kernel risulta condiviso tra tutti i container presenti nell’host, tale azione comprometterà tutti i container e l’host stesso utilizzato dal container engine. Pertanto maggiori sono le applicazioni presenti sull’host, maggiori sono i rischi in termini di negazione del servizio. Attivare un controllo del level patching del kernel avendolo sempre aggiornato è una cosa da implementare nei processi di patching management anche perché, su molti Unix/Like, le patch cadenzate non sempre comprendono le patch del Kernel del sistema ma potrebbero essere patch separate.
2. **Sicurezza dei contenitori:** Attenzione a dove si scaricano, producono, archiviano e versionano i container. Sono già presenti su DockerHub molti container archiviati dagli utenti ma di fatto fallati, contenenti una pila software con vulnerabilità note e malware generico. Il Patching management dei Middlware presenti nei container dovrà essere gestita con cadenza e non cambia rispetto alla basic-virtualization, se un container contiene uno stack “fallato”, tali vulnerabilità potranno essere utilizzate per violare il sistema.
3. **Patching del Container Engine:** sicuramente questo potrebbe essere un reale problema. Essendo l’engine un

middleware del sistema Host, effettuando il patching di questa componente potrebbero esserci problematiche di blocco applicativo che dovranno essere gestite mentre tali exploit potranno essere utilizzati per violazioni delle applicazioni stesse per passare a livello Host. Oggi questi prodotti sono acerbi e poco conosciuti, cominciando ad essere più popolari l'incremento dei CVE/exploit sarà maggiore e quindi dovrà essere valutato il patching a caldo dei container engine per evitare disservizi applicativi oppure prevedere hypervisor di mediation per consentire lo switch dell'applicativo da una VM ad un'altra per evitare il fermo applicativo.

4. **Resource Locking:** i contenitori condividono risorse, qualora l'accesso a tale risorse viene compromesso con un qualche exploit (ad esempio l'impostazione di un file System in Read-Only-Mode oppure CPU Reservation, ecc…) tutti gli altri container non potranno più svolgere il loro lavoro compromettendo l'intera applicazione.
5. **Compromissione delle password:** Come sempre, le chiavi di accesso alle risorse condivise (ad esempio alla base dati) e l'host stesso devono essere sicure, cifrate e mai in chiaro. Questo aspetto ricade nella logica dello sviluppo sicuro del software già discussa nel precedente articolo. Ad esempio utilizzando Database SQL ovviamente condivisi da più container, una stringa di connessione in chiaro e un Database mal configurato comprometterà tutti i dati presenti nel db stesso, ma questa è storia già conosciuta nelle applicazioni monolitiche.

…e come spesso accade, partendo da una semplice architettura potremmo complicarci le cose per implementare un più complesso modello PaaS e quindi combinare hypervisor e Container engine o addirittura utilizzare OpenStack per massimizzarne i benefici, ma direi di aspettare e non pensarci ancora e di farci prima le ossa su quello che adesso risulta disponibile, perché un conto è lo stack di sviluppo (facile e flessibile e massivamente utilizzato oggi) e un conto è montarci una

applicazione enterprise che concentra il business-core della nostra azienda.

Quindi Docker, RkT e container-engine sì ma con la giusta attenzione. Facciamoci le ossa in laboratorio, esploriamo il comportamento applicativo di questi nuovi prodotti e cominciamo a svilupparci sopra piccole applicazioni per poi "bucarle" e comprendere se queste nuove tecnologie in termini di sicurezza realizzano i benefici attesi e solo dopo portare queste nuove tecnologie negli ambienti di produzione prima su piccoli sistemi e dopo su sistemi enterprise.



L'informatica è fatta di stagioni, siamo passati dalla Platform-1 alla 3 passando dall'era del client-server e oggi ai container, ma quando penso ai container non so perché rivedo in piccolo il concetto più allargato di isolamento delle subaree in architettura mainframe e il tutto risulta coerente con quanto diceva un mio vecchio responsabile che l'ingegneria del software e l'informatica in generale è fatta da un input, un programma e un output.

Articolo a cura di **Massimiliano Brolli**

L'articolo Sicurezza nell'era dei container proviene da ICT Security Magazine.

https://www.ictsecuritymagazine.com/articoli/sicurezza-nellera-dei-container/