Rendere sicuro un ambiente Docker Swarm non sicuro

Le principali fonti :

- https://github.com/docker/docker.github.io/blob/master/swarm/configure-tls.md Guida presente su github contenente la configurazione di tls su un cluster Docker Swarm
- https://docs.docker.com/engine/security/https/ Guida dalla documentazione ufficiale di Docker Swarm per rendere sicuro il Docker daemon socket
- https://github.com/docker/docker.github.io/blob/master/registry/deploying.md Guida presente su github per il deploy di un registry
- problemi di carattere inferiore su https://stackoverflow.com/

Tramite le guide abbiamo creato uno script che mette in pratica il seguente procedimento:

Facciamo in modo che uno dei nodi presenti nel cluster, nel nostro caso swarm-3, faccia da Certificate Authority, Questa scelta è stata dettata dalla comodità di generare le chiavi tramite la prima macchina che viene creata. In quanto CA, swarm-3 crea inizialmente il certificato e la chiave per la certificate authority (ca.pem e ca-priv-key.pem); successivamente vengono create una chiave privata e una chiave pubblica per il manager dello swarm (swarm-1), per i nodi worker (swarm-2 e swarm-3) e per il nodo client (dev). Queste chiavi vengono successivamente copiate nella cartella condivisa “resources/certs”

In seguito installiamo le chiavi copiando, tramite uno script dedicato per ogni macchina, certificato e chiave privata della macchina e certificato della CA nella cartella ~/.certs di ogni nodo. La presenza del certificato della CA è necessaria per il riconoscimento del nodo come “sicuro” nella comunicazione TLS.

A questo punto, sempre all’interno degli script dedicati, configuriamo i nodi per comunicare sulla rete accettando solo connessioni che utilizzano TLS sulla porta 2376 (default per la comunicazione TLS) e lo facciamo modificando il file “override.conf” impostando le chiavi da utilizzare per la connessione. Il file “daemon.json” invece viene sovrascritto con un file vuoto (contenente solo le parentesi graffe). Naturalmente dopo la modifica viene ricaricato e riavviato il demone a bordo di ogni nodo.

Creamo il cluster swarm tramite il nodo manager e aggiungiamo i nodi restanti allo swarm.

Per testare il funzionamento delle chiavi dal nodo client (dev) verso i nodi dello swarm (ad esempio swarm-1) abbiamo utilizzato il seguente comando:

```
docker --tlsverify --tlscacert=./.certs/ca.pem --tlscert=./.certs/cert.pem --tlskey=./.certs/key.pem -H=swarm-1:2376 version
```

Questo va a connettere il nodo corrente con quello specificato nell’opzione -H, andando poi a chiedere informazioni sulla versione del demone docker in ascolto sul nodo specificato.

Per quanto riguarda l'utilizzo del protocollo TLS senza dover specificare le chiavi ad ogni comando, è possibile esportare delle variabili d'ambiente che DOCKER_TLS_VERIFY=1 e DOCKER_CERT_PATH=/home/vagrant/.certs/ ma ciò non sembra dare i risultati sperati, continua a essere necessario inserire le chiavi ad ogni comando.

Per quanto riguarda i test di sicurezza sull'utilizzo delle chiavi, come ci si aspettava, provando a togliere le chiavi il demone rifiuta la connessione chiedendo se si sta provando a connettersi senza TLS quando questo protocollo è richiesto. Se invece si tenta di connettersi sulla porta 2375, usata in precedenza, si riscontra un messaggio di errore in forma di domanda che chiede se il demone docker sia avviato su quella porta, dimostrando che è impossibile connettersi in altri modi.

L'utilizzo del protocollo però genera alcuni problemi risoluzione dei nomi, ad esempio specificando l'indirizzo IP del nodo invece del nome si incorre in un errore poiché l'indirizzo IP non viene riconosciuto come valido ai fini della validazione del certificato. Allo stesso modo non vengono accettate connessioni da localhost, ciò impone di specificare le chiavi e l'host di destinazione in qualsiasi comando docker, come se si trattasse di host remoto anche nel caso si volesse semplicemente verificare la versione del demone sul nodo da cui si lancia il comando.

Riguardo al registry, anche in questo caso vengono create una coppia di chiavi da parte della CA che vengono poi installate nella cartella ~/.certs del nodo swarm-1, sul quale viene creato il registry come servizio.
Abbiamo modificato lo script "start-swarm-registry.sh" in modo che modifichi la label del nodo swarm-1 aggiungendogli la presenza di un registry, e poi creando il servizio specificando anche qui le chiavi da utilizzare e la porta 433, default per il servizio TLS.

Se nel file "override.conf" viene lasciata l'opzione --insecure-registry my-registry:433 (modificando la porta da 5000) il registry sembra funzionare ma si va semplicemente ad aggirare il protocollo TLS.

Non siamo riusciti a far andare a buon fine i test sul registry. Quando si tenta di effettuare il push di un'immagine (nei test si è utilizzato ubuntu:16.04) la connessione viene rifiutata in quanto non viene riconosciuta la certificate authority.
Probabilmente nella configurazione del registry bisogna aggiungere in qualche modo un secret con il certificato della CA.

I passi per testare il progetto sono:

1. accedere nella cartella del progetto e poi nella sottocartella vagrant
2. avviare le macchine con vagrant up
3. entrare nella macchina dev tramite vagrant ssh dev (è possibile eseguire l'operazione anche sulle altre macchine swarm-1, swarm-2 e swarm-3)
4. eseguire sudo docker --tlsverify --tlscacert=./.certs/ca.pem --tlscert=./.certs/cert.pem --tlskey=./.certs/key.pem -H=swarm-1:2376 pull ubuntu:16.04
5. eseguire sudo docker --tlsverify --tlscacert=./.certs/ca.pem --tlscert=./.certs/cert.pem --tlskey=./.certs/key.pem -H=swarm-1:2376 tag ubuntu:16.04 my-registry:433/my-ubuntu

6. eseguire `sudo docker --tlsverify --tlscacert=./.certs/ca.pem --tlscert=./.certs/cert.pem --tlskey=./.certs/key.pem -H=swarm-1:2376 push my-registry:433/my-ubuntu`

a questo punto è visibile la risposta “Get https://my-registry:433/v2/: x509: certificate signed by unknown authority” che indica il rifiuto della connessione dato che non viene riconosciuta la certificate authority.