March 2022

DE-CENTRALISED, AUTHENTICATED AND SECURE COMMUNICATION BETWEEN CONTAINERS

Niranjan M M

DE-CENTRALISED, AUTHENTICATED AND SECURE COMMUNICATION BETWEEN CONTAINERS

AUTHOR:
Niranjan M M

ABSTRACT

Containers are becoming more and more popular and are now widely adopted as a new deployment model for virtual, cloud and fog-based applications. In multiple scenarios, probability to expose internal application communication to other unintended applications running on the other containers would increase, which in turn leads to various security vulnerabilities/risks. There are several ways to authenticate and secure the communication as well as data exchange between containers either on a Private or even Public (cloud) network. But none of existing methods provide distributed way to authenticate containers and provide secure communication between the containers. The techniques presented herein propose method to provide trusted and fully distributed authentication model which will be used by containers belongs to the same distributed application to identify, authenticate each other, and provide secure communication.

DETAILED DESCRIPTION

Containers are becoming more and more popular and are now widely adopted as a new deployment model for virtual, cloud and fog-based applications.

For example:

- Multiple products use Kubernetes which is an orchestrator (deployment, scaling, management) for containerised applications.
- Multiple vendors incorporating LXC for running containerised applications on switches/routers etc.,

Since containers (e.g., LXC, Dockers) are mainly a set of processes running in the same/different namespace, they may share lot of resources such as the network stack, the file system, the storage, RAM, CPU etc., Further we are seeing more and more use cases of distributed applications (for scale) across different Data Centers (DC), different fog computing nodes etc., With these

scenarios, probability to expose internal application communication to other unintended applications running on the other containers would increase, which in turn leads to various security vulnerabilities/risks.

As compared to traditional VM model (on ESXi, KVM) or Public Cloud instances (on AWS, GCP, Azure) which provides good isolation (a VM/Cloud instance can be seen as a real host), the container model raises security as well as authentication concerns. Hence, we need a mechanism to provide peer-to-peer authentication between containers. There are several ways to authenticate and secure the communication as well as data exchange between containers either on a Private or even Public (cloud) network. But none of existing methods provide distributed way to authenticate containers and provide secure communication between the containers.

The techniques presented herein propose method to provide trusted and fully distributed authentication model which will be used by containers belongs to the same distributed application to identify, authenticate each other, and provide secure communication.

Also, techniques presented here could be visualized with traditional Blockchain, but considering the drawbacks of blockchain as below, the proposed method uses Holochain. Traditional Blockchain based approach would authenticate each transaction (carrying data/message/record) exchanged between the containers. But this does not scale very well since the time to validate a transaction (by miners) does not allow faster communication between containers.

Considering the drawbacks/limitations of Blockchain:

- Not scalable, as data needs to be replicated on all blockchain nodes and limited by the number of transactions.
- Convergence time is more.
- Validate and adding transactions to the Blockchain takes comparatively more time.
- Time/clock should be in sync among the Blockchain nodes as timestamp is part of transaction and used during merging of ledgers.

Holochain overcomes all the above limitations of blockchain and provides fully distributed peer-to-peer authenticated and secure communication.

Also, rather than authenticating each transaction of Holochain, it would be optimal to authenticate a "KEY" (also called "session key") which is having TTL (valid for a given period).

This key can be used to encrypt/decrypt the data exchanged between containers (security) or can be used for signing the data (authenticity). This security or authentication mechanism can be applied at the application level, network packet level or container group level.

This method provides trusted and fully distributed authentication model which will be used by containers belonging to the same distributed application to identify and authenticate each other using Holochain technology. This method leverages the fully distributed ledger characteristic of Holochain DHT and provide a secured mechanism to exchange data between containers. Holochain uses self-consensus and does not need validator/miners as in Blockchain. Holochain also provides Privacy by having Validation Rules (VRs) and these VRs could be unique per distributed application hosted on multiple containers.

The technique presented herein is explained in step-by-step as below:

A. Prerequisites:

I. Holochain enabled containers:

- Each container running one or more processes of distributed application.
- Containers are enabled with Holochain functionality (by running Holochain application, called, Happ)
- Consider sending data/packet/record/message as a transaction (let us say T1).
- Validation Rules are setup according to policies defined, also called as "Policy Attributes". These validation rules can be used to provide Privacy among containers and can be maintained per distributed application running on different containers deployment.

II. Holochain application (Happ):

- Happs will have validation rules (also called as Holochain DNA) and would be the first entry in the local Hash-chain.
- Only validation rules require global consensus (compared to blockchain where every transaction need global consensus).
- With these validation rules as the foundation, each node keeps an immutable record (transaction) of their own actions on a local Hash-chain.
- Each Hash-chain entry (record/transaction) is cryptographically signed to prove authorship and ensure accountability. Transactions are mutually counter signed by both the peer containers. They can audit each other's chains before agreeing to the transaction.

- Mutually signed transactions are updated to the Holochain DHT by containers running processes belongs to the same distributed application. To have optimization, only meta data of the transaction can be maintained in DHT, anyway their local hash-chain have the complete details of the transaction.

B. Peer-to-peer communication between Containers:

- Containers would be running one or more processes related to the same distributed application.
- Instead of authenticating every transaction, would be creating a "KEY" (also called "session key") along with TTL as transaction and mutual authenticating this among peer containers.
- The KEY creation would be done as explained below:
    - One of the containers would be selected as "Key Server" based on the Container ID, UUID, lowest/highest IP/MAC address etc.,
    - Use HKDF (HMAC Key Derivative Function) to generate unique KEY on the selected "Key Server".
    - The "Key Server" uses this generated KEY as part of transaction.
    - "Key Server" also selects TTL value based on the application requirement (can be configurable).

1. Container-1 would create a transaction T1 (with KEY and TTL value)
2. Container-1 would sign the transaction T1 (using its Private Key SK).
3. Container-1 sends the signed Transaction T1 to the Container-2.
4. Container-2 validate the transaction T1 received from the Container-1 (using Public Key PK of Container-1).
5. Container-2 also sign the transaction T1 before adding to the Local Hash-chain (now it is mutually signed).
6. Container-2 adds the mutually signed transaction T1 to its Local Hash-Chain.
7. Container-2 sends the mutually signed transaction T1 to the Container-1.
8. Container-2 publish/send mutually signed transaction T1 to the shared Holochain DHT.
9. Container-1 adds mutually signed transaction T1 to its local Hash-chain.

10. Container-1 also publish/send mutually signed transaction T1 to the shared Holochain DHT.

Figure-1 depicts the above steps, where-in Container-1 communicate with Container-2 using Holochain. Even though Container-3 and Container-4 instantiated on the same Host, they will not be able to read/modify/repeat/tamper the data exchanged between Container-1 and Container-2.

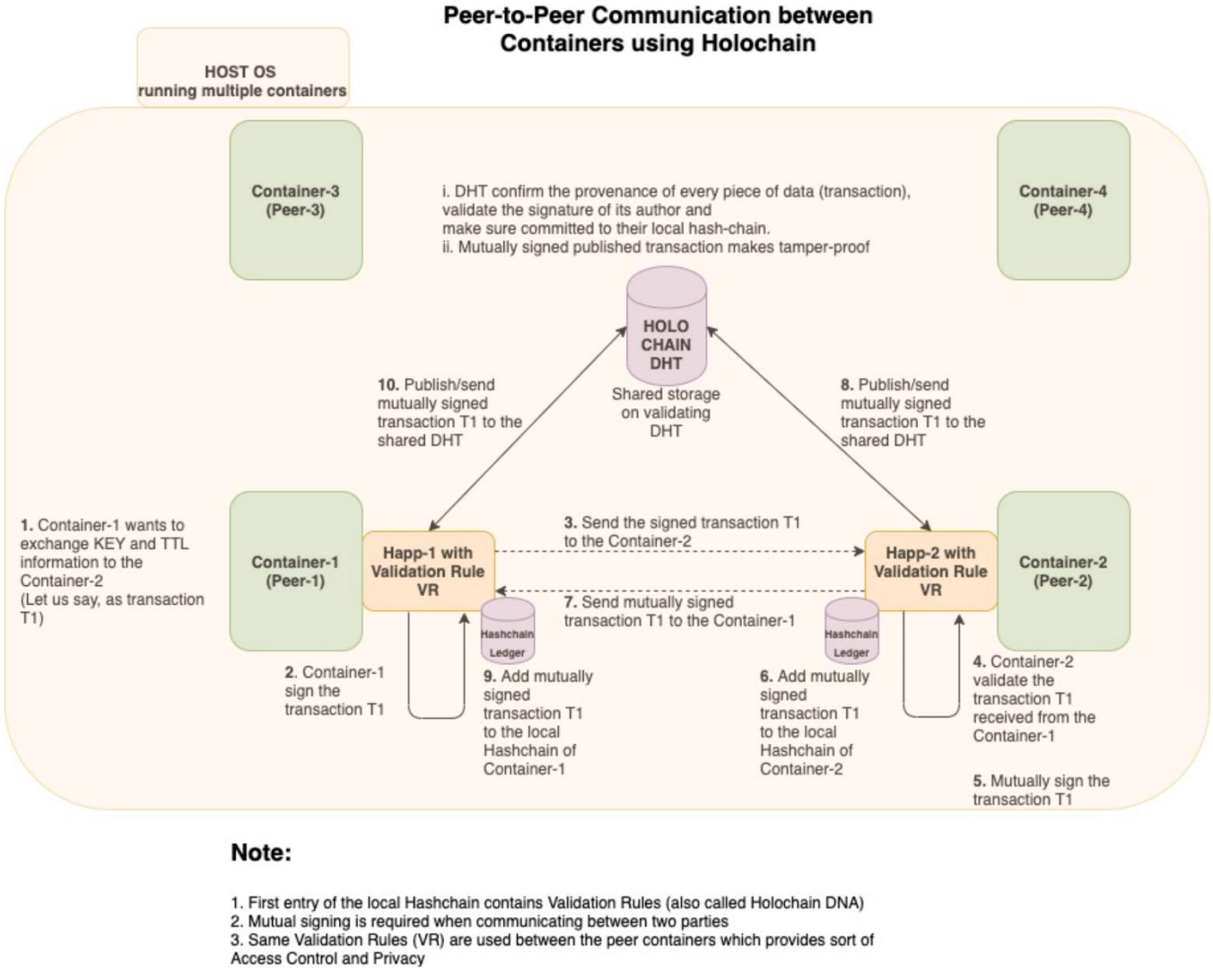

Figure-1

C. Authentication between Containers:

As in step (B), "KEY" and its TTL value are exchanged securely and trusted way (immutable) between container using Holochain (self-consensus, full distributed) technology. Hence authentication between the container is achieved.

D. Encryption between Containers:

The "KEY" exchanged between containers can be used to encrypt/decrypt the data exchanged between containers. Upon TTL expiry (or after 80% of TTL value), containers exchange renewed "KEY" and TTL value as explained in step (B).

E. Multi-container deployments and secure channel:

- It is also possible to create transaction with KEY and TTL value along with Validation Rules, whereby multiple containers can agree to create a meshed secured communication network between each other. (Similar to mesh communication or multicast).
- The transaction itself can contain all the necessary information (protocol, algorithm etc.,) to establish an explicit secured communication channel between themselves (e.g., VPN, IPSec, TLS etc.,)

Notes:

- Fully distributed and self-consensus - There is no central point managing secure communication between containers. It is almost impossible to compromise the system integrity.
- Using Holochain to exchange KEY information between containers is a new approach.
- Instead of authenticating every transaction, it would be optimal to mutually authenticate "KEY" between the containers. "KEY" is generated using HKDF on one of the containers ("Key Server").
- With fully distributed approach, it is up to each distributed application running on multiple containers to establishing/revoking the secure communication channel.
- Since Holochain DHT stores cryptographically signed transaction (metadata), it is possible to track authorship and ensure accountability.
- The transaction exchanged between peer containers can be different depending on some peer characteristics, allowing for different level of security when required.

The techniques presented herein propose to use self-consensus (without the need for an external validator, as required in blockchain based methods) to provide authenticity between containers. Holochain has in-built access control and privacy by having Validation Rules (VRs), which can be used to enhance Privacy in containerised deployments. Holochain is a scalable architecture as compared to blockchain and hence this method inherently solves scalability issues. This method is generic enough that, it can be incorporated in different containerised models such as LXC, Kubernetes, Dockers etc., This method is suitable for privacy sensitive deployments (enterprise, healthcare). Additionally, huge container deployments such as Data Center, SDN etc., can use this method.