Corda Node Deployment

2018-02-19

v1.0

1. Introduction

Deploying Corda on-premises within a bank presents a number of specific security challenges. In particular there is typically a strongly-layered approach to defence. At a minimum there will be 3 separate zones: DMZ (directly addressable from the Internet), Production (not addressable from the Internet), and Data (no access to the Internet).

DLT/Blockchain platforms present a significant new deployment challenge because they exhibit behaviour similar to a peer-to-peer network; they both initiate and terminate connections with external hosts on the same port. This type of behaviour is new in many enterprise deployments.

This document describes the approach to deploying within these environments using R3 Corda.

2. Threat Categories

There are many potential threats within an enterprise network, but these largely fall into the following categories:

- Threats against assets held by systems inside the enterprise environment.
- Threats against the systems within that environment.
- Denial of service threats.

2.1. Threats Against Assets

Threats against assets are classified as potential attack vectors against systems that control access to individual assets. This class of attack is typified by an attacker finding means to send messages to a victim system that manages an asset that will result in the asset being accessed or modified when it should not.

2.2. Threats Against the Network Environment

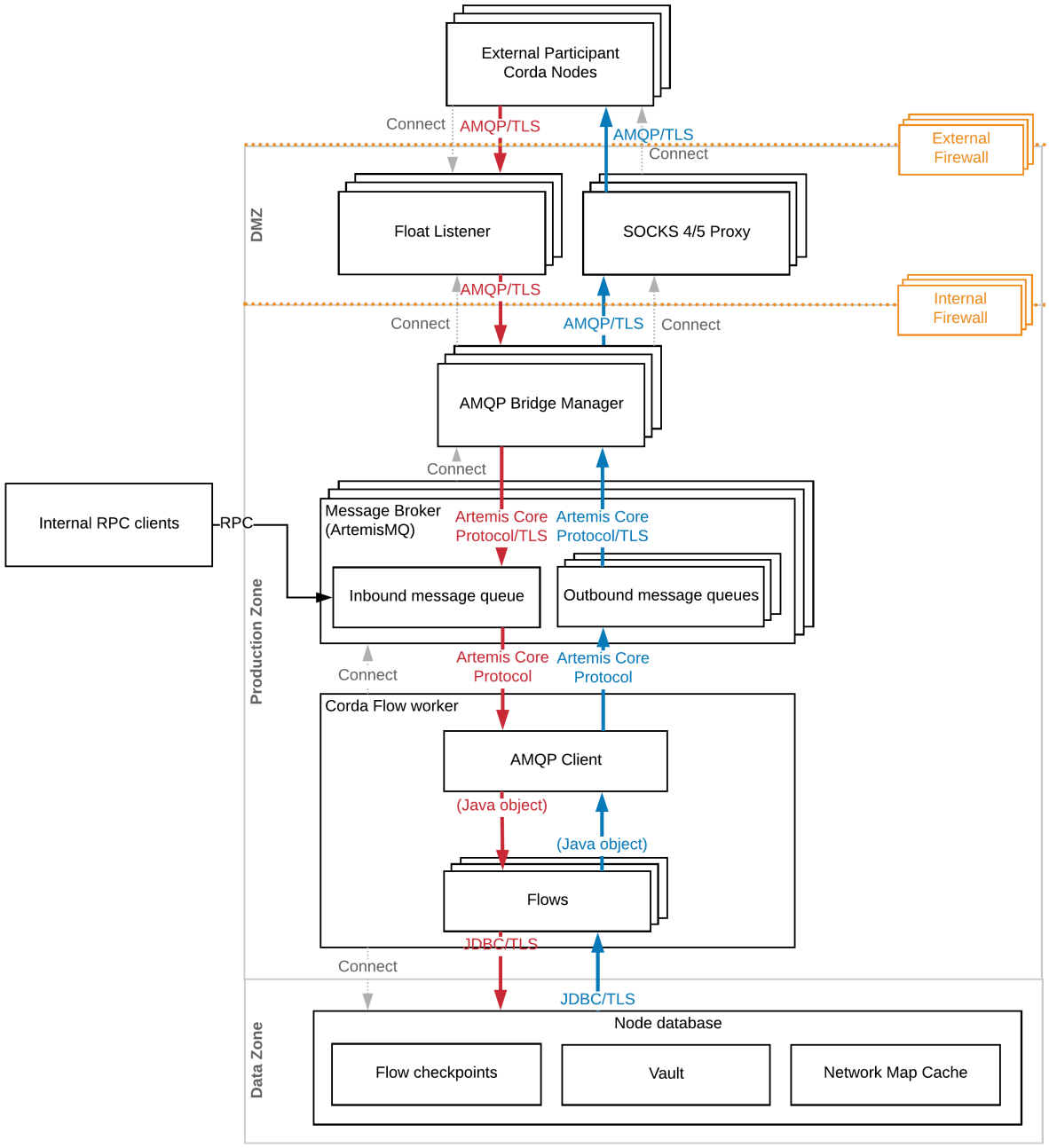
Threats against the network environment are classified as potential attacks systems to which a message was not apparently intended. This class of attack is typified by an attacker finding means to send messages to some victim system other than the one apparently intended as the recipient

2.3. Denial of Service Threats

Denial of service threats are broadly classified as attempts by other systems to prevent a local system from performing its intended purpose. This class of attack is typified by an attacker finding a way to send messages that fully consume some resource within a victim system. For example, this might be an attack that saturates connection tables, CPU availability, storage capacity, memory availability, networking bandwidth, etc.

3. R3 Corda Proposed Node Deployment

R3 Corda has been designed to address concerns around deployability. It adds a new component specifically to meet enterprise security requirements. The node architecture is depicted thus:



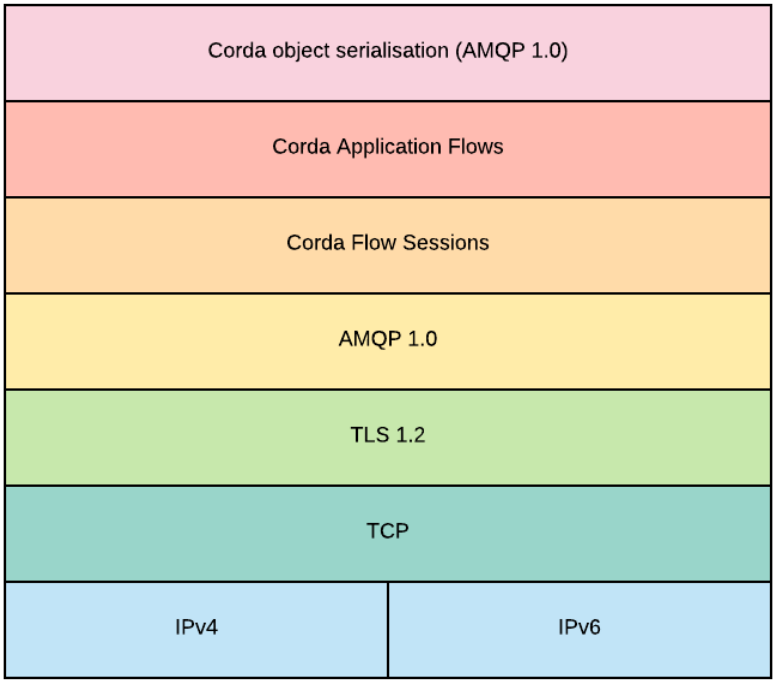
The diagram shows three significant characteristics. All of the components are connected over TCP. The blue arrows depict the egress flows of information from the local Corda node to external nodes, while the red arrows show the ingress flows of information from external Corda nodes to the local node. The grey arrows indicate the direction in which connections are initiated. A key feature is that there are no connections initiated from the DMZ to the production zone, ensuring that the production zone is always in control of any information flow.

In a separation of responsibilities, there are separate communication components in the ingress and egress path, although the SOCKS 4/5 proxy is, strictly, not necessary for the correct operation of the system.

Note that the SOCKS 4/5 proxy must be provisioned with the TLS certificate of the local node, so it may not be possible to use a pre-existing proxy, unless it has the capability to offer certificates based on the originating system IP address.

3.1. Protocol Description

Corda node-to-node communication follows a well-defined binary protocol. The layering is depicted thus:



TLS 1.2 provides per-connection encryption. AMQP 1.0 provides a standard message queue infrastructure. The Corda flow sessions and application flows are provided by Corda, while data structures serialised by Corda are encoding using the same format as the AMQP 1.0 message queue layer.

All layers of the protocol stack are binary-encoded, as opposed to being text-based. Text-based protocols are notorious for enabling implementation weaknesses to be exploited by allowing for imprecision around the size of messages and their exact interpretation. By comparison Corda requires very precise encoding of messages in order to guarantee that serialised forms are identically deserialised as any recipient, since otherwise the hash digests of such deserialised data would not match those of the serialised forms (an essential characteristic)

3.2. Egress Data Path

Egress messages are generated by the local Corda node and forwarded through the internal firewall to a SOCKS 4/5 proxy. This proxy acts as a protocol break at the TLS layer since the TLS sessions from the local Corda node to external nodes are different. The SOCKS proxy terminates the connection from the local node and re-establishes a new connection to an external node.

Note: The TLS connection from the local Corda node to the proxy is optional, given the potential need to perform content scanning prior to the SOCKS proxy.

3.3. Ingress Data Path

Ingress messages originate with a remote Corda node and are forwarded to the firewall float component within the DMZ. The firewall float terminates the TLS sessions from the remote node, using the TLS certificate provided to it by the local Corda node. The firewall float also acts as a protocol break as the ingress messages are held by the float until they are subsequently requested by the local node.

Note: The TLS connection from the firewall float to the local Corda node is optional, given the potential need to perform content scanning following the firewall float.

The ingress data path is the one primarily concerned with most of the threat models. We consider these below.

3.3.1. Threats Against Assets

Corda takes very great steps to ensure that underlying assets are protected against protocol-level attacks. Message encoding is a binary format, and thus much more precisely defined than text-based protocol designs. This, in turn, means that any structured content fields are precisely delineated and interpreted. Where any fields are presented to the vault database they are automatically escaped to prevent the possibility of SQL injection attacks. No content is simply “passed through”.

Another class of potential attack against assets might be that requests were incorrectly formed or validated, but Corda requires that all transactions are not only well-formed, but also correctly signed by appropriate counterparts. Most importantly, Corda’s view of the correctness of transactions is not only defined by external requests, but also that the local node agrees that any such request is also correct from a business logic perspective. A request deemed invalid by the local node will be logged and rejected.

3.3.2. Threats Against the Network Environment

Corda’s use of AMQP and the firewall float means that there is no possibility of traffic ingressing to the DMZ being forwarded to any system in the production zone without the explicit control of the local Corda node. Traffic is not forwarded beyond the firewall float, and only the Corda node will pull data from the firewall float. This eliminates a number of potential attack vectors in which ingress traffic is scheduled such that it will be automatically forwarded to an unexpected host within the production zone.

3.3.3. DoS Threats

Corda requires that all TLS sessions are mutually authenticated by both endpoints, and a requirement to participate within a Corda network is that both derive their root identity from a well-defined certificate issuer (root of trust). Actors outside of the Corda network will not have this and will thus see incoming connections rejected. Should an actor inside of the Corda network misbehave in a way that has DoS-like characteristics then it will quickly be detected, and the originator of the problem will rapidly be known by virtue of logging events. As each identity is issued by a common authority, then it is easy to determine which node, and thus which legal entity, is responsible.

3.4. Logging of Protocol Events

Given that Corda makes use of a binary protocol, some of the text-based tools used to check protocol correctness are unlikely to be useful. As binary protocols are more precisely defined, however, it is relatively easy for the various components that decapsulate protocol encoding to identify well-formed, or mal-formed messages. Where mal-formed messages are detected then such events should be logged, along with details of the originating node/nodes, so that these can be escalated to any operations teams monitoring Corda traffic.

Logging should take place at both the firewall float and the Corda node.

R3 will provide details of the logging events.