JXTA security in basic peer operations

J. Arnedo-Moreno[1] and J. Herrera-Joancomartí[2]

Abstract—**JXTA is an open-source initiative that allows to specify a set of collaboration and communication protocols which enable the creation and deployment of peer-to-peer (P2P) applications. This paper provides a survey on its current state regarding the topic of security. The study focuses on the security evaluation of standard peer operations within the JXTA network, highlighting which issues must be seriously taken into account in those applications sensitive to security.**

Index Terms—**peer-to-peer, security, peer group, analysis, evaluation, distributed systems, JXTA, Java.**

I. Introduction

Peer-to-peer environments provide a virtual network where all involved parties collaborate in order to supply basic services, such as content sharing, processing or messaging. It is also assumed [1] that all peers have equivalent capabilities, and a central server with more processing power is no longer necessary, ensuring a high degree of decentralization and autonomy of participants. Such environments have become highly popular in recent times due to its great potential to scale and the lack of a central point of failure.

Just as the popularity of peer-to-peer applications has risen, so has concerns regarding their security, specially since it is no longer possible to trust a central server which capitalizes all security operations. As peer-to-peer applications move from simple data sharing (such as Gnutella [2] or Bittorrent [3]) to a broader spectrum, they become more and more sensitive to security threats and it becomes capital to take into account which measures exist in current peer-to-peer platforms before deploying them.

JXTA [4] (or "juxtapose") is a set of open protocols that enable the creation and deployment of peer-to-peer networks. JXTA protocols enable peer-to-peer applications to discover and observe peers, enable communication between them or offer and localize resources within the network. Such protocols are generic enough so they are not bound to a narrow application scope, but are adaptable to a large set of application types. For that reason, they also keep implementation independence, so they can be deployed under any programming language or set of transport protocols.

In this paper, a survey of the current state of security in JXTA for basic peer operations is provided. Such operations are not analyzed in an isolated way, but the whole peer life cycle is taken into account. The results of this study may benefit security-aware platform developers and designers which want to create JXTA applications, by providing them a detailed list of which issues must be taken into account. JXTA users may also benefit by realizing which may be the limitations of their applications on the scope of security, so they may take additional measures in order to guarantee a completely secure environment.

The paper is organized as follows. In section II, an overview of the JXTA platform is provided in order to understand its main characteristics and methods of operation. Following, in section III, the basic evaluation model is described by categorizing basic peer operations and threats under the JXTA model. Section IV presents the security analysis. The final conclusions are outlined in section V.

II. An overview of JXTA

This section provides a general overview of the main JXTA concepts and components in order to understand the peer operations explained in section III and their security concerns. A detailed explanation of JXTA can be found in [5], [6].

The fundamental JXTA architecture is divided in three distinct layers, as shown in figure 1.

The *Core* layer contains the minimum and essential characteristics, common to all peer-to-peer networks. This includes peer discovery and communication (even when behind firewalls or NAT) and peer creation, as well as the basic security services.

The *Services* layer includes all network services which are not absolutely necessary, but provide desirable capabilities such as resource search and indexing, as well as resource storage and sharing.

The top layer is the *Applications* one, which includes the applications deployed with the use of the JXTA framework, such as instant messaging, file sharing or content management.

Notice that the distinction between services and final applications is not always clear, since what a client may consider an application may be considered a service by another peer. For that reason, the system is designed in a modular way, letting developers chose the set of services and applications which most satisfy their needs. All JXTA components are within these three layers.

A. Peers

Each peer in the JXTA virtual network is identified by a unique Peer ID, operating in an independent and asynchronous manner regarding other peers. However, some dependencies may exist depending on which roles they partake.

Usually, peers will act as *edge peers*, which could be considered the standard peer type in any desktop application on most devices. They implement the JXTA core and standard

(1) Estudis d'Informàtica, Multimèdia i Telecomunicacions, Universitat Oberta de Catalunya, Rb. Poble nou 156, 08018 Barcelona. jarnedo@uoc.edu.

(2) Escola Tècnica Superior d'Enginyeria, Universitat Autònoma de Barcelona, Campus de Bellaterra. jherrera@deic.uab.cat

This work was partially supported by the Spanish MCYT and the FEDER funds under grant TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 "ARES", funded by the Spanish Ministry of Science and Education.

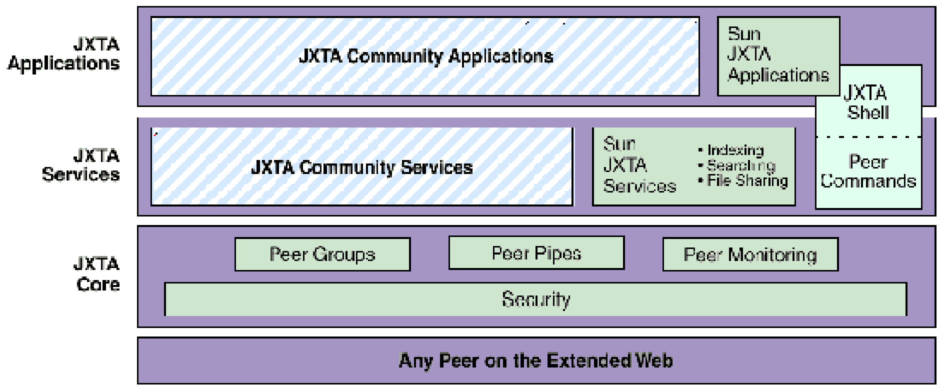

Fig. 1. JXTA layered architecture

services layers as shown in figure 1 and may interact with any JXTA protocol.

Edge peers may also partake two additional roles in order to avoid some specific network constraints: *minimal* and *proxy*. This decision will usually depend on its hardware or bandwidth capabilities.

Devices with specific resource constraints (memory, CPU) may act as *minimal peers*, which only implement the JXTA core layer. They are usually simple devices such as sensors or domotics. In order to use any necessary service to operate within the network, they must rely on *proxy peers*. A proxy peer summarizes and answers requests on their behalf. Any edge peer may partake the role of a proxy peer.

A very important role is that of *rendezvous peers*, which maintain a global index of available resources and help other peers find network services. They are also used as beacons which newly connected peers may use in order to join the network. For that reason, rendezvous peers will usually be well-known ones, with a DNS name or a static IP address.

Additionally, *relay peers* provide routing information and are used in order to store and send messages between peers separated by firewalls or NAT.

As we can see, a JXTA network may vary between a pure and a hybrid peer-to-peer model (with rendezvous peers acting as some kind of super-peers) depending on which role peers finally partake.

B. Protocols

As explained, JXTA defines a set of protocols (six, specifically) which enable the deployment of peer-to-peer networks. Using these protocols, peers may collaborate in a fully autonomous manner by publishing and discovering available resources within the network. Peers may also cooperate in order to route messages, allowing full communication, without the need for them to understand or manage different network topologies.

All JXTA protocols are asynchronous and based on a request/response model, which means that for any given request, zero, one or many responses may be received.

- The *Peer Discovery Protocol (PDP)* allows peers to publish their own resources and make them available to other peers. Any kind of peer may send PDP messages. This protocol is the default discovery protocol, but it is possible for applications to implement and deploy their own protocols.
- In order to obtain information regarding to other peers, the *Peer Information Protocol (PIP)* is used. Using this protocol, it is possible to query peer capabilities or monitor its behavior.
- Peers use the *Peer Resolver Protocol (PRP)* in order to send requests to one or several peers and manage responses. The PRP protocol uses an unique ID is associated to every request, which is included in each message. Other core protocols, such as PDP, make use of PRP in order to create its own requests.
- The *Pipe Binding Protocol (PBP)* establishes virtual communication channels between peers, acting as abstract endpoints above any physical network.
- Routes between peers are found with help of the *Endpoint Routing Protocol (ERP)*. Whenever a peers is a about to send a message to a destination but does not know the path, a ERP message is sent to other peers asking whether they know a path.
- Finally, the *Rendezvous Protocol (RVP)* is responsible for the efficient propagations of messages within a group of peers, allowing peers to connect to services (exchange messages). RVP is used in turn by PRP.

C. Resource publication

Any kind of resource available within the JXTA network, including peers, peer groups, pipes or services, is described by an *advertisement*. All advertisements are XML documents containing an unique ID and all information regarding that resource and how it may be accessed and exchanged between peers using the JXTA protocols. Peers cannot access a resource without previously retrieving its associated advertisement. Every peer maintains a local cache where all received advertisements are stored for a later use. The local cache directly makes use of the file peer system in order to organize its content (directories and files).

A sample advertisement is shown in XML listing 1.

XML Listing 1 - Peer Advertisement

```
<xs:element name="PA" type="jxta:PA"/>

<xs:complexType name="PA">
  <xs:sequence>
    <xs:element name="PID" type="jxta:JXTAID"/>
    <xs:element name="GID" type="jxta:JXTAID"/>
    <xs:element name="Name" type="xs:string" minOccurs="0"/>
    <xs:element name="Desc" type="xs:anyType" minOccurs="0"/>
    <xs:element name="Svc" type="jxta:serviceParams"
                minOccurs="0" maxOccurs="unbounded"/>
  <xs:sequence>
</xs:complexType>
```

Any peer may publish an advertisement in order to announce that some resource is available by using two different methods: local and remote publication.

In *local publication*, the advertisement is indexed and stored in the peers local cache. Following, the advertisement's index is pushed to a rendezvous peer and is then distributed and replicated between all rendezvous in the global super-network peers using the Shared-Resource Distributed Index (SRDI) service [7], [8]. The rendezvous network acts as a remote index cache.

By using this method, it is possible for peers outside the local network (out of broadcast range) to retrieve group advertisements by asking a rendezvous peer. It also enables peers which where off-line for some time to retrieve advertisements published during its disconnection. Whenever a peer receives the advertisement, it is indexed, stored in a local cache and assigned an expiration date.

The advertisement retrieval mechanism is outlined in figure 2.

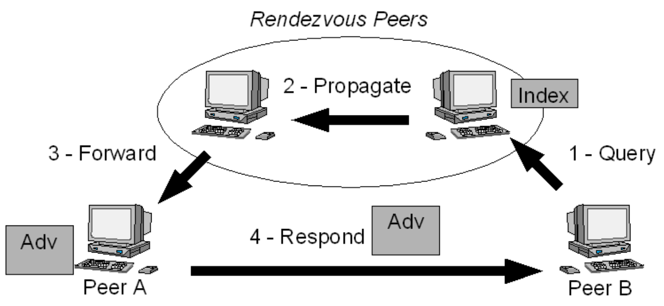

Fig. 2. Advertisement retrieval from Rendezvous peers

It must be remarked that during the publication process, the original advertisement is always kept in the peers' local cache, only its index is distributed. This means that in case the peer goes offline, the advertisement will become unavailable. That makes sense, since also the resource the advertisement publicizes will be unreachable.

In *remote publication* not only indexes are distributed, but the full advertisement itself via the JXTA propagation mechanism. This method is useful in case that the advertisement must be reachable even when the publishing peer is offline. However, under the remote publication method, no assumptions can be made about which peers will really store the advertisement and for how long.

In both methods, when the expiration date is reached, the advertisement is considered stagnant and flushed from the cache, unless the same advertisement is received again, which renews its expiration date. Advertisements may be periodically retransmitted in order to attain permanency or update parameter changes.

As can be seen, advertisement publication and discovery are very important steps in the JXTA peer operation.

D. Messaging

JXTA peers use *pipes* in order to exchange messages and access available services. JXTA messages are XML documents with ordered message elements which may contain any type of payload. Messages are the basic data exchange unit in JXTA and all protocols are defined as a set of messages exchanged between peers.

Pipes provide an asynchronous, unidirectional and unreliable communication channel by default. However, bidirectional reliable channels may be provided on top of them. They offer two operation methods: *unicast* pipes, which allow one-to-one communications, and *propagation* pipes, which allow one-to-many.

JXTA pipes are an abstraction and are not bound to a specific IP address or port. They have a unique ID and are published just like any resource in the JXTA network, so any peer may update them whenever its physical location changes. Both input pipes (message reception) and output pipes (message sending) are considered *pipe endpoints*, the actual destination in the physical network. Endpoints are dynamically bound to peers via the PBP.

E. Peer groups

JXTA [4] introduces the concept of *peer group*: a collection of peers with a common set of interests. This concept is one of the main foundations of the JXTA architecture and is prevalent throughout all its specification [6]. Offering the possibility to create different (but not necessarily disjoint) groups of peers operating under the same overlay network allows to segment the network and offers a context to peers for publishing and accessing different services.

Peer group boundaries provide a secure framework in order to grant or deny access to the offered services. Peer groups form logical regions whose boundaries limit access to group resources, in a way similar to a VPN [9], but operating at the application layer. Other interesting uses are the ability to provide a scoping or monitoring environment, where different classes of traffic and advertisements are limited to peer group members.

Peer groups are published, discovered and accessed just like any other resource in the network, via advertisements.

In order to allow peer group management, JXTA defines the basic primitives for group membership and access control: the Membership and the Access Services. Both are core services which make use of the base JXTA protocols specification in order to achieve their ends. Just the primitives are defined, specific applications may implement their own Membership and Access services depending on their needs.

The Membership Service manages identities within a peer group, providing each group member a *credential*. Peers may include this credential in messages exchanged within a peer group in order for each other member to know who is making a request. With this information, the JXTA Access Service may evaluate the credential when a service is accessed and decides whether the request will be granted or denied.

A peer establishes its credential within a peer group by successfully *joining* it. Before a peer may join a group, it must be authenticated by providing a correct *Authenticator* to the Membership Service. An Authenticator contains all the required information in order for the Membership Service to check that the requested identity can be granted. Each different Membership Service specification provides its own definition of an Authenticator, suited to its needs and inner workings.

The join process is divided in three distinct steps:

- In the *setup* step, the peer chooses which authentication method will be used for the whole process. If all parameters are correct and the choice is feasible, the peer receives an Authenticator from the Membership Service.
- Following, in the *application* step, the peer completes the Authenticator with all necessary information and tests its correctness. It will not be possible to join the group until the Authenticator is correctly initialized.
- Finally, in the *validation* step, joining the group is possible if the Authenticator is correct. The Membership Service checks whether the peer may assume the claimed identity and creates a credential.

In case that a peer decides to give up membership to a specific group, it is possible to *resign*. When this happens, the credential is discarded. Group resignation is voluntary, the Membership Service does not support active membership revocation triggered by other members.

The credentials generated in the join process may be sent to the whenever a group service is accessed, as part of the protocol exchanges. The JXTA Access Service provides mechanisms in order to check them, allowing services to decide whether access should be granted or not.

The Access Service provides a single primitive in order to check a credential for a privileged operation. The possible results are disallowed (access denied), permitted (access granted), permitted but expired (the operation would be permitted but the credential has expired) and undetermined (unrecognized credential).

III. Security evaluation model

The first step in order to assess the security degree in JXTA applications is to identify which is the standard peer lifecycle, so that it becomes clear which are the most common operations and, consequently, which deserve better attention on regards to security.

Once such operations have been identified, a list of usual security concerns in P2P environments is provided. Our security evaluation model will be based in the cross-reference of standard operations and such threats, in order to evaluate how the system is protected against each one.

A. Standard peer operation cycle

In this section, the standard peer operations for a peer participating in a JXTA network are described. The order in which they are presented is a logical one for most scenarios. However, it must be taken into account that such order may vary depending on peer role (except, obviously, platform startup).

1) Startup platform: This is the initial step in order to setup the platform in the physical device which will hold a JXTA peer. This process mainly consists in loading the required libraries and creating the necessary data structures for network connectivity.

At startup, all peers automatically join a default bootstrap peer group named *netPeerGroup*. This peer group is well known to all peers, since it's ID is hard coded into the platform distribution. Peers may stay in this group or decide to leave once they are connected to the JXTA network. At this stage, edge peers may also try to reach relay or rendezvous peers depending on their local configuration.

2) Join a peer group: A peer will join those peer groups formed by peers it wants to interact with. This step will usually be the next, following startup, so all later operations are only within the boundaries of those specific peer groups and not the whole network. The peer will locate the peer group advertisement and join it via its Membership Service. In the case that such peer joins the group for first time, it must locate the peer group advertisement via PDP. Otherwise, the advertisement will be stored in its local cache.

A peer may join several groups, which means that this operation may be performed several times.

3) Publish own resources: Any resource that the peer holds and wants to make available to the rest of peer group members will be announced by creating and publishing an advertisement as described in section II-C. This step includes announcing its own presence, by publishing a peer advertisement, or creating a new group, by publishing a peer group advertisement.

4) Discover other resources: Available resources in peer groups which it has joined are discovered by retrieving their advertisements via PDP. This includes discovering other peers and pipes in order to initiate message exchanges. Usually, pipe advertisements, necessary in order to initiate message exchange, are embedded into other more generic advertisements such as service advertisements.

5) Exchange messages: This would be the most frequent operation during any peer operation cycle. Once the resource to be accessed is located, an outbound pipe is opened and messages are exchanged across it. In the case that it is the peer the one who is offering the service, an inbound pipe is opened in order to process incoming messages.

6) Disconnect: The peer resigns from all peer groups and goes to offline state in a tidy manner.

This list of operations maintain some degree of abstraction, but each one actually represents a set of more basic steps. In Table I, a breakdown of each operation into such basic steps is provided. A more detailed explanation can be found in [6]. Nevertheless, from a security assessment standpoint, the chosen degree of abstraction is enough to provide a clear

Startup platform	Load platform Join *netPeerGroup* Open network listeners Open local cache
Join a peer group	Locate group advertisement Instantiate group Fill in Authenticator Join
Publish own resources	Create advertisement Local publication Remote publication
Discover other resources	Locate peer advertisement Locate pipe advertisement Store advertisements in local cache
Exchange messages	Open pipe Send messages Receive messages Check Access Service
Disconnect	Close connections Shutdown platform

TABLE I
BASIC OPERATION SUBSTEPS

idea of which are the possible scenarios during any peer's full operation cycle, from startup to disconnection.

B. Security threats in P2P networks

The standard security threats in the traditional client/server environment still hold good in P2P environments. Furthermore, the P2P paradigm shift introduces new concerns that must be taken into consideration when designing P2P frameworks. The move form a passive stance (client) to an active one (peer) in the network easily propagates such concern across all its members. Security attacks in P2P systems are classified into two broad categories: passive and active [10].

Passive attacks are those in which the attacker just monitors activity and maintains an inert state. The most significant passive attacks are:

- *Eavesdropping*, which involves capturing and storing all traffic between some set of peers in the serach for some sensitive information (such as personal data or passwords).
- *Traffic analysis*, where the attacker not only captures data but tries to learn more by analyzing its behavior and looking for patterns, even when its content remains unknown.

In active attacks, communications are disrupted by the deletion, modification or insertion of data. The most common attacks of this kind are:

- *Spoofing*, in which one peer impersonates another, or some outside attacker transforms communications data in order to simulate such an outcome.
- *Man-in-the-Middle (MitM)*, where the attacker intercepts communications between two parties, relaying messages in such a manner that both of them still believe they are directly communicating. This category includes on route data alteration.
- *Playback or replay*, in which some data exchange between two legitimate peers is intercepted by the attacker in order to reuse the exact data to make it look like a real exchange. Even if message content is encrypted, such attacks can succeed so long as duplicate communications are allowed and the attacker can deduce the effect of such a repeat.
- *Local data alteration*, which goes beyond the assumption that attacks may only come from the network and supposes that the attacker has local access to the peer, where he can try to modify the local data in order to subvert it in some malicious way.

Apart from security threats that take into account a malicious attacker, it is also very important to take into account *software security flaws* in a security survey. Specifically, which steps are taken by the developers in order to minimize the probability that a bug or an optimistic assumption in the development process may later jeopardize the system at deployment.

IV. SECURITY EVALUATION

From the standpoint of basic security requirements which are desirable in JXTA, they are very similar to those of any computer system: confidentiality, integrity and availability. In order to achieve them, these requirements should translate into an architecture that includes authentication, access control, audit, encryption, secure communication and non-repudiation.

JXTA remains neutral to cryptographic providers or security schemes. In its initial conception it does not mandate any specific security solution, providing a generic framework where different approaches can be plugged in. Enough hooks and place holders are provided in order for each specific application to implement its own security solution. Nevertheless, in a present day peer-to-peer framework, relying in the fact that each application will build from scratch its own security solution is not enough, since it will usually mean that security will be overlooked, as its is often the case. It is very important that the basic tools and functionalities are already there, providing a default degree of security but allowing further modularization if necessary. As such, basic security services (encryption, integrity and authenticity) should be provided at the core layer, even though some applications may chose not to use them.

In this section we will analyze whether the current iteration of JXTA (version 2.5) is up to this desiderata for each peer basic operation and is ready for the standard threats to peer-to-peer networks.

A. Startup platform

Since during the framework startup the networking capabilities are not operative yet, no threat related to a networked environment applies. However, at this precise moment JXTA libraries are loaded into the system, then it does make sense to take into consideration software flaws and local data alteration on such libraries from a local attacker. This part of the analysis will also take into account those aspects related with application deployment.

JXTA is an open source software (OSS) project, which is a good indicator when specifically analyzing security [11].

As its is well known, security through obscurity does not work, and any software design which depends upon secrecy is guaranteed to fail, since secrets have a way of getting out. Since JXTA code is public [4], it has been audited by a large number of individuals all across the Internet. The use of an OSS approach not only ensures current security, but allows direct improvement from the JXTA developer community, maximizing the networking effect.

Nevertheless, it is worth mentioning that opening the source code creates the opportunity for individuals to review security, but it cannot guarantee that such reviews will occur. There is also the fact that no guarantee can be made that a review will find any particular security flaw in a system, but that problem is common to closed source projects. In any case, OSS allows developers with security concerns to directly assess whether the JXTA framework is up to their needs.

On the other hand, because of its OSS nature, JXTA distributions can be easily modified (since complete build instructions are readily accessible), which means that it is trivial for an attacker with some coding expertise to create a malicious version of the libraries. In order to avoid malicious distributions, the official JXTA project page protects builds with SHA1 digests [12]. A local attack is necessary to actually deploy a malicious distribution. However, current distributions offer no mechanisms in order to detect that library integrity has been compromised, with a relatively easy solution such as code signing [13]. External tools would be necessary in order to ensure this kind of control, which is out of the scope of this survey.

B. Join a peer group

A necessary step in order to join any peer group lies in retrieving its peer group advertisement. As the implications of this specific substep will be explained in subsection IV-C, here we will focus on the actual group instantiation and join operation.

As described in subsection II-E, the membership Service is the key security measure in the group join operation. Through this service, peers claim identities by proving its ownership. Even though this service is defined as generic in the specification, in order for each application to implement it according to its own needs, the JXTA reference implementation, as far as version 2.5 [5], provides three available Membership Services which are ready to use.

The *None* Membership Service is intended for peer groups which need no authentication. Since any peer may claim any identity, it is recommended that credentials should only be used within the group for purely informational purposes. This service is widely used in applications with no nearly security concerns.

The *Passwd* Membership Service relies on a Unix-like username and password pair for peer authentication. In order to claim an identity, the correct password must be provided. The list of pairs (username and password) is distributed to all group members, which means that the password file equivalent roams freely through the overlay network. This group membership service was created as a sample and a means of testing, since it is completely insecure. For that reason, it is advised in the JXTA documentation that it should never be used in any serious application.

The default Membership Service is *PSE*, which stands for *Personal Security Environment* and is the only model which the developers consider secure. As such, it is the only one we will consider for the security analysis.

The PSE service provides credentials based on PKIX [14] certificates, as described in XML listing 2.

XML Listing 2 - PSE Credential XML schema

```
<xs:complexType name="jxta:PSECred">
  <xs:sequence>
    <xs:element name="PeerGroupID" type="jxta:JXTAID"/>
    <xs:element name="PeerID" type="jxta:JXTAID"/>
    <xs:element name="Certificate" type="base64binary"
             minOccurs="1" maxOccurs="unbounded"/>
    <xs:element name="Signature" type="base64binary"/>
  </xs:sequence>
</xs:complexType>
```

The authentication procedure in order to join a PSE peer group can be summarized as follows:

1) User introduces its personal password.
2) Peer initializes Peer Group Authenticator with that password, Peer ID.
3) With that information, an encrypted keystore in the local cache is located and opened.
4) User's certificate chain is retrieved.
5) That certificate chain becomes the group credential for that Peer.
6) User may interact with other ones in the same Peer Group.
7) Private key in keystore is used at leisure by Peer when needed in secure protocols.

All the enumerated steps in the join process for the the PSE Membership Service are completed via local calls to JXTA libraries. For that reason, peer group joining is not concerned with network-based attacks (eavesdropping, traffic analysis, MitM or replay), since there is no real network operation. it also means that any threat on the join process must be exploited by a local attacker.

As we can see, three actors interact in this process: the final User (the human being in front of the computer screen, or some agent), the Peer (the application) and the Peer Group (JXTA libraries which control group access). That means that two methods of spoofing must be taken into consideration:

- Impersonating the User: Unauthorized access to keystore content. This is equivalent to taking control of another user's peer.
- Impersonating the Peer: Unauthorized identity claim and credential generation. This is equivalent to being able to claim any identity within a peer group.

In the first case, all security relies on the strength of keystore encryption and its password. Unfortunately, the keystore is stored as a simple file, which may be easily copied and distributed, and no mechanisms exist in order to plug in more advanced methods of key management (such as hardware cryptographic tokens).

As far as Peer impersonation is concerned, then it is very interesting to point out the implications of the fact that peers under a PSE Membership Service are authenticated only by being able to access a local keystore. During the process, the Membership Service is not concerned with the validity of certificate chains (signed by a proper Certification Authority or not expired), and the certificate content is never checked. As a result, anybody with access to a private key and a certificate (a self-signed one is sufficient) will be able to correctly join any group using PSE and claim any identity. For that reason, the default join scenario is easily threatened by spoofing attacks on the Peer, since anybody may create a valid keystore using public domain tools [15]. There is no real security on peer identity claims.

As presented, PSE is more of a kind of toolbox that allows the implementation of different models based on securing identities via digital certificates, since it provides no clear structure of how trust is managed in a peer group (whose signatures are trusted and which peers are allowed to sign certificates). In order for PSE to properly function, the application must agree on which are the real trust anchors (PGP-like trust chains, a centralized CA) and some method in order to guarantee key authenticity must exist (solving this is left up to each application). When each peer may generate certificates, PSE may not work, since it is not possible to easily guarantee key authenticity. This is a weakness the developers of JXTA agree it should be adressed [16]. Even under this assumption, it must be remarked that correctly setting a trust anchor in a pure peer-to-peer environment is no easy feat, since in-band certificate generation procedures are easy prey to MitM attacks.

Nevertheless, because of the way the join operation works, any peer will still be able to claim any identity and get some group credentials. It will be during the message exchange operation, as credentials are actually checked, when the false identity will be detected. Those peers which hold the necessary information in order to generate a correct PSE Peer Group Authenticator, but are not really group members because their certificate is not properly signed are named *interlopers*. In this scenario, they can become an annoyance, but are easily spotted and dealt with.

Finally, we should mention that the choice of a trust anchor is not an easy task when dealing with a peer-to-peer network if peer equality must be preserved.

C. Discover and publish resources

Resources provided by peers in the JXTA network are represented by XML documents named advertisements, as explained in subsection II-C. In order to discover such resources, its advertisement must somehow be retrieved. Advertisement discovery and retrieval is achieved via message exchange using the PDP and PRP core services (see subsection II-B). Since it is a network-borne operation, we can focus on all threat types. We will discuss both resource discovery and publication in this section, since both share the same security mechanisms (only data flow direction changes between both operations)

In the current reference implementation, advertisements may be secured by digitally signing them at application level, using a special type of advertisement named *Signed Advertisement*. By using digital signature directly on the advertisement, it is possible to support both local and remote publication (via propagation to multiple peers).

In order to use this special type of advertisement, it is mandatory to previously join a group that use the PSE Membership Service, since the necessary cryptographic keys in order to generate and validate the signature are obtained from its associated keystore and credential. Signed advertisements will only be sent to members of that group. In the case of that peer joining peer groups which do not use this membership service, signed advertisements will not be exchanged between group members. As a result of this, any concern related to PSE is inherited by signed advertisements, such as the lack of real authenticity without setting a trust anchor at application level.

The XML schema definition for a Signed Advertisement is shown in Listing 3. It contains the signer's credential (the `PSECred` element, a credential for a PSE Membership Service), the signature and the original advertisement.

XML Listing 3 - Signed Advertisement XML schema

```
<xs:element name="SA" type="jxta:SA"/>
  <xs:complexType name="jxta:SA">
  <xs:sequence>
    <xs:element name="PSECred" type="jxta:PSECred"/>
    <xs:element name="jxta:Signature" type="base64binary"/>
    <xs:element name="jxta:Advertisement" type="base64binary"/>
  </xs:sequence>
</xs:complexType>
```

The Advertisement element encapsulates the original XML advertisement as plain text encoded via the Base64 algorithm [17]. The content of the `Signature` element is generated by applying the RSAwithSHA1 algorithm to the original advertisement, XML formatted (not its Base64 encoded form). In order to feed the algorithm, the XML data is processed as plain text. The result is henceforth Base64 encoded in order to be represented as plain text into the XML document. Once a signed advertisement is received by a peer, it's signature is validated, the actual advertisement desencapsulated and the stores in the local cache.

Currently, as far as advertisements is concerned, JXTA does not seem concerned with passive attacks, since it offers no advertisement protection against them. They are freely exchanged between peers in plain text. In fact, since they are structured using XML, it is very easy for an eavesdropper to read search for specific content (no need to process binary structured data). A human being can directly interpret advertisements with a text editor.

No effort is made either in order to masquerade advertisement traffic, so it is feasible that an attacker may obtain some interesting information by just analyzing traffic, specifically detecting which peers offer more resources (since they are the ones which publish more advertisements). Using this method, it is possible to search for interesting peers when looking for potential victims to attack.

In fact, since anybody may instantiate any group acting as an interloper, as remarked in the previous subsection, and then discover advertisements bound to it, this kind of attacks are easy to perform. It is not even necessary to tap the network. How easily advertisements are exchanges is both a bonus for open services and a bane for for tight security environments.

If we assume that applications which use PSE correctly set a trust anchor in order to guarantee certificate authenticity, then active attacks such as spoofing, MitM and replay attacks may be correctly countered by digitally signed advertisements when discovering resources. Using this method, false advertisements may still occur within the peer group, but because of non-repudiation, it will be easy to pinpoint offenders.

However, as far as resource publication is concerned, since every peer is completely reliant on its rendezvous peer in order to properly distribute the advertisement index to the rest of the network, and no control is made about which peer may become a rendezvous one, it is possible to pull off MitM attacks by masquerading as a one. No control mechanisms exists in order to automatically detect a misbehaving rendezvous. For that reason, each application should always deploy some method in orders for peers to be able to identify real rendezvous peers.

Finally, since secure advertisements lose the signature when stored into the local cache, the threat of a local attacker still exists, since it is possible to modify the local cache content, inserting or modifying false advertisements which redirect service access peer to malicious nodes.

D. Exchange Messages

Since core service level messaging was already explained in the previous section, only message exchange related to application services is analyzed here: end-to-end transport using JXTA pipes.

The JXTA specification guarantees end-to-end transport security via two different protocols: TLS (Transport Layer Security [18]) and CBJX (Crypto-Based JXTA Transfer [19]). Both protocols provide different flavors of security: TLS provides private, mutually authenticated, reliable streaming communications, whereas CBJX provides lightweight secure message source verification (but not privacy).

TLS relies on PSE credentials in order to properly process the authentication handshake, which means that it can only be used within a PSE peer group. XML messaging is directly sent over TLS sockets.

CBJX is a JXTA-specific protocol which uses digital signature in order to provide integrity and authentication. It adds an additional information block to the secured message, as shown in XML listing 4: a `PeerCert` element, which contains the source peer certificate, both the source and destination addresses, and the source peer ID. Both the message body and the cryptographic information block are digitally signed, generating two separate signatures. The certificate inside the cryptographic information block is used to validate both signatures.

In order to generate both signatures, XML data is serialized, processed as plain text, and fed to the signature algorithm.

Apart from digital signature, CBJX provides lightweight authenticity by using Crypto-Based Identifiers [20] (CBIDs). The concept of CBIDs, or statistically unique and cryptographically verifiable IDs (SUCV IDs), was initially conceived for IPv6 addressing in order to solve the issue of address ownership, avoid router supplantation attacks and binding update packet spoofing [21], [22]. Using this mechanism, each address is automatically bound to a specific node. It is important to notice that by using this method authentic messaging is provided without the need of certificates issued by a trust anchor under a PSE membership service, in contrast to nearly all security mechanisms previously described.

XML Listing 4 - CBJX crypto-information XML schema

```
<xs:complexType name="cbjx:CbJxMessageInfo">
  <xs:sequence>
    <xs:element name="PeerCert" type="base64binary"/>
    <xs:element name="DestinationAddress" type="string"/>
    <xs:element name="SourceAddress" type="string"/>
    <xs:element name="SourceID" type="jxta:JXTAID"/>
  </xs:sequence>
</xs:complexType>
```

By combining both CBJX and TLS, it is possible to trump both passive and active attackers by achieving data privacy, integrity and authenticity. Application developers may decide which protocol to use depending on their constraints (such as a non-PSE peer group).

However, in both cases (TLS and CBJX), information security is only provided during transit by protecting the JXTA transport protocol at a lower layer. Once the transport encapsulation is removed and information is stored into the local peer, it is no longer secured. It also must be taken into account that both types of transport methods do not support full advertisement propagation, they only support end-to-end communications. That means that applications which are based on multicast are still prone to security threats.

An additional step to be considered during message exchange when accessing a service is checking peer credentials in order to decide whether some peer has real access to that service. This is necessary since, as we could see in subsection IV-B, actually anybody may instantiate a peer group and try to access resources. This may be achieved in JXTA by using the peer group Access Service.

As far as the Access Service is concerned, the current JXTA reference implementation offers three kinds of access control, each one bound to each different membership service credential type:

The *Always* Access Service, which does not really check for access control and allow any operation. It is the default Access Service for peer groups.

The *simpleACL* Access Service uses Access Control Lists in order to establish which identities may perform the different group operations. The access lists are distributed as parameters within the peer group advertisement.

The *PSE* Access Service provides an interface to PKIX certificate path validation. A trust anchor is set for the validation process and all credentials are validated against this anchor in order to decide whether the the operation is permitted or not.

It must be pointed out that the current approaches to the Access Service are strictly tied to ensuring that some identity may access some service. Whether that identity really belongs to the peer group is never checked, it is always assumed correct. Since the membership service is not up to the task of checking group membership either (any peer may claim any identity), as exposed in section IV-B, this is something JXTA developers should take into account.

In addition, the access service provides a single primitive which just checks credential content, but does use on any kind of authentication protocol. This is not sufficient to guarantee protection against spoofing, since credentials are freely exchanges across the network (they are public). Some other method must exist which tests credential authenticity (such as TLS or CBJX) in order to guarantee authenticity.

E. Disconnect

No real vulnerabilities threaten the disconnect operation, apart from those which force an unintended shutdown due to unauthorized local access to the application. However, there is little the application could do, since we move to issues with the operating system, so it can be considered outside the scope of this study. The disconnect operation was mainly included for the sake of completeness in formalizing the peer's full lifecycle.

V. Conclusions and further work

As an OSS project, JXTA has been intensely reviewed, and as a result, its security features have improved over time. It can be summarized that the current implementation of JXTA has evolved to include an acceptable level of security, fulfilling minimum requirements for present day applications. However, this is at the cost of being bound to a very specific group membership model (PSE). In the case that a custom model is used in some application, we will find out that most of its security capabilities may no be directly used (only CBJX). This is not always desirable in a framework that was conceptualized to be open and easily adaptable to any environment. It would be useful that any custom application security model could make use of as many as possible JXTA secure mechanisms such as TLS or advertisement signature.

An additional feature that could be interesting, constrained by the assumption that PSE will always be used, would be the capability to use different types of keystores, apart from that stored in the local cache. Specially, being able to go beyond using the file system as cryptographic storage.

It is also important to take into account when designing JXTA applications that, even though PSE provides a certificate-based secure environment, it is still necessary to chose some method in order to guarantee key authenticity. PSE assumes no trust model, just provides the necessary tools in order to deploy it.

Finally, JXTA still has some gaps pending to be filled even then all its security capabilities are used at their best. First of all, no current mechanism exist in order to secure messaging for propagation mechanisms (specially one that provides a some degree of privacy). In addition, no security exists for the local cache, even though very important data is stored inside. At least some degree of integrity would be desirable (such as maintaining advertisement signatures).

Further research includes providing basic security services at core level without the need of PSE. Our main efforts will be twofold. First of all, creating a peer group environment where membership is really checked. Then, providing advertisement security, such as authenticity and confidentiality, in cache storage as well as transport.

References

[1] Andrew Oram, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[2] "Gnutella", http://rfc-gnutella.sourceforge.net.

[3] B. Cohen, "Incentives build robustness in bittorrent", *1st Workshop on the Economics of Peer-2-Peer Systems*, 2003.

[4] Sun Microsystems, "Project JXTA", http://www.jxta.org.

[5] "Jxta 2.5 rc1", June 2007, http://download.java.net/jxta/build.

[6] Sun Microsystems Inc., "Jxta v2.0 protocols specification", 2007, https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.

[7] M.Abdelaziz M.Duigou C.Haywood J.C.Hugly E.Pouyoul B.Yeager B.Traversat, A.Arora, "Project jxta 2.0 super-peer virtual network", Tech. Rep., SunMicrosystems,Inc, May 2003.

[8] Abdelaziz M. Traversat, B. and E. Pouyou, "Project jxta: A loosely-consistent dht rendezvous walker", Tech. Rep., SunMicrosystems,Inc, March 2003.

[9] Huston G. Ferguson, P., "What is a vpn?", Tech. Rep., Cisco Systems, 1998.

[10] Brookshier D. Krishnan N. Govoni D., Soto J.C., "Jxta and security", *JXTA: Java P2P Programming*, pp. 251–282, 2002.

[11] Landwehr C. Caloyannides M, Written B., "Does open source improve system security?", *Software IEEE*, vol. 18, no. 5, pp. 57–61, 2001.

[12] NIST, "Fips pub 180-1: Secure hash standard", 1995, http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[13] SUN Microsystems Inc., "Jarsigner", http://java.sun.com/j2se/5.0/docs/tooldocs/windows/jarsigner.html.

[14] CCITT, "The directory authentication framework. recommendation", 1988.

[15] SUN Microsystems Inc., "Keytool", http://java.sun.com/j2se/5.0/docs/tooldocs/windows/keytool.html.

[16] Yeager B., "Enterprise strength security on a jxta p2p network", *P2P'03: Proceedings of the 3rd Interantional Conference on Peer-to-Peer Computing*, p. 7, 2003.

[17] Ed. S. Josefsson, "Ietf rfc 3548 - the base16, base32, and base64 data encodings", 2003, http://www.ietf.org/rfc/rfc3548.txt.

[18] Allen C. Dierks, T., "Ietf rfc 2246: The tls protocol version 1.0", 1999, http://www.ietf.org/rfc/rfc2246.txt.

[19] D. Bailly, "Cbjx: Crypto-based jxta (an internship report)", pp. 108–109, July 2002.

[20] Castelluccia C. Montenegro, G., "Crypto-based identifiers (cbids): Concepts and applications", *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 97–127, 2004.

[21] T. Aura, "Cryptographically generated adresses (cga)", http://www.ietf.org/rfc/rfc3972.txt.

[22] Luciano Bononi and Carlo Tacconi, "Intrusion detection for secure clustering and routing in mobile multi-hop wireless networks", *Int. J. Inf. Secur.*, vol. 6, no. 6, pp. 379–392, 2007.