Security analysis of JXME-Proxyless version

Marc Domingo-Prieto, Joan Arnedo-Moreno
Estudis d'Informàtica, Multimèdia i Telecomunicació
Universitat Oberta de Catalunya
Barcelona, Spain
{mdomingopr, jarnedo}@uoc.edu

Jordi Herrera-Joancomartí
Escola d'Enginyeria
Universitat Autònoma de Barcelona
Campus de Bellaterra, Spain
jherrera@deic.uab.cat

***Abstract*—JXME is the JXTA specification for mobile devices using J2ME. Two different flavors of JXME implementation are available, each one specific for a particular set of devices, according to their capabilities. The main value of JXME is its simplicity to create peer-to-peer (P2P) applications in limited devices. In addition to assessing JXME functionalities, it is also important to realize the default security level provided. This paper presents a brief analysis of the current state of security in JXME, focusing on the JXME-Proxyless version, identifies existing vulnerabilities and proposes further improvements in this field.**

I. Introduction

Peer-to-peer (P2P) networks allow peers to provide and consume services in a collaborative way. Examples of these services are content sharing, processing and messaging. In this kind of network, it is assumed that all peers have equivalent capabilities, as well as a high degree of decentralization and autonomy.

P2P technology, that has been widely used in traditional wired network environments, is now moving to the mobile paradigm [1] since new wireless technologies are becoming more available (WLAN, 3G, 3.5G,...) and more powerful hand-held devices (like smart phones or mobile Internet devices, MID) have been developed. However, the massive deployment of P2P mobile applications may depend on the tools available for developing such applications in a transparent way.

There are different platforms that allow programmers to develop mobile P2P applications ([2], [3], [4]). One of these platforms is JXME [4], a set of open protocols specifications that enables the creation and deployment of P2P networks over mobile devices. The advantage of JXME, in front of other proposals, is that it is the mobile version of the well known JXTA platform [5]. JXTA is a set of protocols which allow peers to communicate, publish and find resources, and consume remote resources, independently of the actual transport layer and the implementation language. JXME allows mobile devices to create a mobile JXTA network and also to participate in a "wired" standard JXTA network. JXME heavily takes into account the idiosyncrasies of mobile devices such as power and storage limitations, and for that reason research has focused on these features [6]. However, security is a very important issue that has been often forgotten in JXME research.

The main goal of this paper is to analyze the security mechanisms that JXME provides. Such analysis should allow to determine which will be the minimum security features included in P2P mobile applications developed on top of JXME. We based our study in JXME-Proxyless, one of the two available JXME versions, since it is the most complex one and the one where peers are actually self-organized. The security analysis performed in this paper follows the idea of [7] where a generic JXTA security survey has been presented. Applying the same methodology, security is not analyzed by reviewing basic peer operations in an isolated manner, but taking into account the whole peer life cycle. With this approach, it is possible to identify the available security mechanisms and how they operate.

The paper is organized as follows. Section II provides an overview of the JXME project. Section III presents the security analysis of JXME-Proxyless. Section IV provides a brief comparison between both versions of JXME. Finally, Section V outlines the conclusions and further work.

II. Overview of JXME

Currently, only Java implementations of JXME exist. They are direct offshoots of the generic JXTA specification, thus sharing many characteristics with the desktop version. A detailed explanation of JXTA's generic protocols and services can be found in [8], however, we will briefly outline the most important concepts.

In both cases, JXTA and JXME, the architecture is completely based on the concept of *Peer Groups*, sets of peers with common interests which agree on shared services. Peer Groups are managed by the *Membership Service*, one of JXTA's core services. Once a peer has joined a Peer Group, any resource may be shared with other group members by distributing its associated *Advertisement*, an XML metadata document describing the resource properties and how it may be accessed. Advertisements are located and distributed using the *Discovery Service*. A network resource cannot be accessed without previously recovering its associated Advertisement. Every time an Advertisement is retrieved by a peer, it is stored in the local cache and assigned an expiration date. At that date, the Advertisement will be automatically flushed. Once a resource has been located, messaging may begin using JXTA *pipes*, abstract endpoints which provide an asynchronous unidirectional communication channel.

Therefore, the Java implementation of JXME can be viewed as a JXTA compatible platform for resource constrained de-

vices, based on the framework specifications for Java ME: *Connected Device Configuration* (CDC) and *Connected Limited Device Configuration* (CLDC). The CDC specification uses the C-Virtual Machine (CVM), an optimized version of the Java Virtual Machine (JVM) [9], it contains some of the standard Java packages, and it is addressed towards high end mobile devices, such as powerful PDA's and smart phones. In contrast, the CLDC specification uses the Kilobyte Virtual Machine (KVM) [10], which has few of the standard Java packages, thus being suitable for lower end devices with very slow processors and very reduced memory. CLDC is further divided into two profiles which define its operation mode: *Mobile Information Device Profile* (MIDP) and *DOcomo JAva* (DOJA). The former is a specification for the usage of Java on embedded devices and the latter is a Java environment specification for DoCoMo's i-mode mobile phone.

Using JXME, any CDC/CLDC device can participate in the JXTA network and exchange messages with any other peer. Unfortunately, because of the limited capabilities of mobile devices, they cannot fulfill some of the JXTA peer basic functions such as encoding JXTA messages in XML, maintaining a local copy of the network state and listening to incoming network information at socket or datagram level. Two distinct versions of JXME currently exist, each one suitable for a different set of mobile devices. On one hand, the *JXME-Proxied* version is a very simple implementation for limited devices, which delegates all the heavy work to an external super-peer, the *Relay Peer*. On the other hand, the *JXME-Proxyless* version is a more complex one, where mobile peers may directly interact with the JXTA network.

In this paper we focus in the JXME-Proxyless version. We consider it is the most interesting one, since it is the most complex and the one where peers are actually self-organized. However, we will provide insights on JXME-Proxied in Section IV.

A. JXME-Proxyless version

The JXME-Proxyless version currently holds the newest and most complete implementation of JXME, having been expected by the community for years. This version is the nearest one to the JXTA specification, allowing mobile devices to directly participate into the JXTA network by themselves, without the need of an external super-peer. Figure 1 shows the JXME network architecture and how it interoperates with a desktop JXTA network. However, the most advanced functionalities of desktop JXTA, such as the Shared Resource Distributed Index (SRDI) [11], have been implemented as lighter versions, taking into account the limited capabilities of mobile devices.

Any peer using JXME-Proxyless is named a *Proxyless Peer* and is able to perform the following actions by itself:

- Discover other devices and services.
- Publish Advertisements about it's own resources.
- Establish direct connections to any other peer.
- Create/join private virtual domains (Peer Groups).

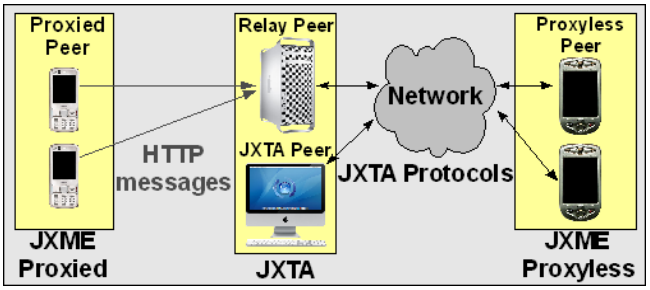

Fig. 1. JXTA and JXME network architecture

- Directly exchange/access content with other Peer Group members.

Proxyless Peers may use JXTA's most important components: Peer Groups, Advertisements and Pipes. Since Proxyless Peers may directly interact with other peer group members under a secure context, Peer Groups are necessary to maintain JXTA's architecture. Advertisements are encoded using XML, just like in the desktop version, in order to maintain language independence. Finally, in spite of its complexity, Pipes are available since direct TCP communications are supported.

However, Proxyless Peers have some limitations on regards to the JXTA base architecture. First of all, even though most JXTA services are implemented, some of they are not, and even when a particular service exists, it must be taken into account that it may not have full capabilities. An obvious example is the Membership Service, which does not support all implementations available in desktop JXTA. Furthermore, another constraint is that a Proxyless Peer cannot act as a super-peer in a JXTA network. As a result, since super-peers help network management, such as caching Advertisements to accelerate search queries, a JXTA network formed only by Proxyless Peers may have scalability issues.

Finally, it is also important to point out that, since Proxyless Peers directly participate in the JXTA network (forwarding messages, finding routes, saving Advertisements, etc.), they have to pay a cost in resource consumption, such as battery power, even when the mobile device is in standby mode.

B. Related work

Some research exists on JXTA, enhancing its basic features [11] and security [7], but not many efforts have been made for JXME specifically.

In [12] an analysis about JXME functionality is found, concluding that JXME-Proxied is not suitable for MANET environments because of its centralized architecture but JXME-Proxyless can fit well in this type of environments. A framework for mobile devices optimized to MANET networks which is compatible with JXTA protocols is developed in [13]. In [14] a framework to allow JXME devices to use bluetooth is presented. This framework permit devices to overcome Bluetooth limitations, such as the maximum number of interconnectable devices and the maximum transmission range.

JXME has also been analyzed and used to build a set of applications. For instance, in [15] JXME-Proxyless is used to implement a distributed collaborative platform which makes people in distributed spaces ubiquitous collaborate with friends and colleagues.

However, regarding security, to our best knowledge, there is no attempts to identify nor improve the JXME-Proxyless security properties.

III. JXME-Proxyless security assessment

Guaranteeing a minimum security level should be one of the main goals in most of the current P2P applications, even though this level may differ depending on the particular needs of each application. In this section, a security analysis of JXME-Proxyless is made in order to evaluate the security level currently provided by the platform. This analysis follows the methodology used in [7], where the general peer life cycle is examined rather than isolated peer actions.

The standard JXME-Proxyless general operation cycle can be summarized in the following steps [7]: Platform startup, Peer Group joining, Resource discovery and publication, Message exchange and Disconnection. A brief description of each step follows:

1) **Platform startup**: This is the first action performed by a JXTA Peer and consists in loading the required libraries and initializing the system prior to going online.
2) **Peer Group joining**: At this step, the peer joins a Peer Group, so interaction with other Peer Group members is possible. Peer Group joining is managed by the Membership Service, one of JXTA's core services, which allows peers to claim unique identities within a Peer Group.
3) **Resource discovery and publication**: Encompasses the distribution and location of Advertisements and how to access it. This action is performed via JXTA's Discovery Service.
4) **Message exchange**: This is the most frequent action in Proxyless Peers, consisting of data exchange, usually in order to access resources, such as available services. This exchange can only exist between same group members and is accomplished using JXTA pipes.
5) **Disconnection**: Peer cleanup before exiting the JXTA network. This is the last action a peer performs before going offline.

A. Attacks in P2P networks

In order to perform a security assessment, it is useful to identify and categorize the most common attack types in P2P networks. All attacks can be divided into two distinct groups, according to the degree of involvement of the attacker [16]: passive attacks, where the attacker just monitors peer activity and network traffic, and active attacks, where the attacker purposely interferes with network activity. Each group can be further classified according to the particular action performed by the attacker.

We are interested in the following attacks:

Passive attacks:

- *Eavesdropping* (Evs): Searching, in message exchanges, for sensitive information such as passwords.
- *Traffic analysis* (TAn): Analyze traffic data, looking for patterns and relevant peers.

Active attacks:

- *Spoofing* (Spf): Impersonating another peer.
- *Man-in-the-middle* (MitM): Intercepting the communications between two parties, transparently relaying forged messages to each one.
- *Playback* (Pb) or *Replay* (Rp): Capturing messages so they can be reused at a later time, simulating a real message exchange initialization.
- *Local data alteration* (LDA): Modifying local data to corrupt the system behavior.
- *Software Security Flaws* (SSF): Exploiting vulnerabilities due to bugs in the source code trying unexpected actions on the software.

B. JXME-Proxyless Security evaluation

From the peer operation cycle and the identification of possible attacks, it is possible to provide a structured security assessment. To identify which vulnerabilities exist, we have designed and performed some attacks which try to subvert JXME operations. Our analysis is focused in active attacks, since they need technical knowledge about the JXME architecture, and rely on active operations to exploit vulnerabilities. Passive attacks are more generic and can be performed using common tools, such as sniffers [17].

1) Platform startup: The first action a Proxyless Peer performs consists in loading the JXTA libraries and creating the default network manager. This operation does not perform any network activity, and thus is protected from external interference at this level. The only vulnerabilities that exist are those related to library authenticity. Since no mechanisms are provided to differentiate a good JXME-Proxyless distribution from a malicious one, it is possible to subvert the system via *local data alteration* attacks.

To prove this flaw, we have designed an attack where an original Proxyless Peer (P_1) tries to send messages to a hacked Proxyless Peer (P_2), who uses a modified JXME-Proxyless library. The attack consists on removing the content of the *publish* and *remotePublish* methods inside the *net.jxta.impl.discovery.DiscoveryServiceImpl* class. Both methods are used to publish and propagate Advertisements to other peers. Therefore P_2 is not able to distribute his Advertisements over the JXTA network. These changes make P_2 unreachable from P_1 and from the JXTA network, since its Peer Advertisement, needed by P_1 or any other peer to route messages to him, is never published.

2) Peer Group joining: The step of joining a Peer Group is handled via the JXTA Membership Service. This is one of JXTA's core services, which manages the group members' identities within the group context. Identities are assigned by successfully completing an authentication process prior

to actually joining the group. The Membership Service is defined as generic in the JXTA specification, leaving up to developers to implement their own version, with the security level required by their applications.

Even though JXTA provides some reference implementations for the Membership Service, JXME-Proxyless provides none at all, allowing any Proxyless Peer to create and join any Peer Groups. Since no Membership Service is implemented, no security really exists for the join operation, and no authentication process is enforced, allowing any peer to claim any identity within the system. We test this security flaw by running two Proxyless Peers that execute a demo chat application provided within JXME-Proxyless library. Both peers exchange messages inside a created new Peer Group. However, we have created an additional peer, who can join this new group, claiming the identity he wants and send messages to those peers inside the group.

3) Resource discovery and publication: Inside JXTA and JXME-Proxyless, resources are published across the JXTA network by distributing an Advertisement. The JXTA specification defines Advertisement security at two distinct levels: at Advertisement level and during its transport. In the former, the secure layer data is directly included in the Advertisement as additional content, whereas in the latter, the Advertisement is processed as a simple message. Security at message layer will be discussed in Section III-B4.

As far as Advertisement level security is concerned, JXME does not provide any security mechanism. They are transmitted without any kind of privacy over the network, thus becoming vulnerable to *eavesdropping* attacks, as well as *traffic analysis*, since an attacker can identify important Proxyless Peers (those with many resources) by the amount of published Advertisements.

Furthermore, we have designed and performed a *Spoofing* attack on Advertisement exchanges, where there are two Proxyless Peers (P_1 and P_2) exchanging messages, and a malicious Proxyless Peer (P_3) trying to impersonate P_2. The structure of this attack is shown in Figure 2 and follows the steps:

1) P_2 publishes his Peer Advertisement, containing his route address.
2) Since no mechanism is provided to authenticate the peers, P_3 can publish a Peer Advertisement using P_2's identifier but adding P_3 address. Once this Peer Advertisement is propagated across the network, it will replace the original P_2 Peer Advertisement.
3) Before P_1 can send a message to P_2, it has to ask for P_2 Peer Advertisement to the super-peer.
4) Super-peer sends P_1 the last P_2 Peer Advertisement.
5) P_1 tries to send a message to P_2, but he will actually send it to P_3 instead. This attack will be reverted when P_2 republishes his Peer Advertisement.

Moreover, still exists a vulnerability inherited of JXTA. In JXTA, super-peers are responsible for the propagation of Advertisements over the JXTA network. However there are not any mechanisms to identify malicious ones. Therefore, a

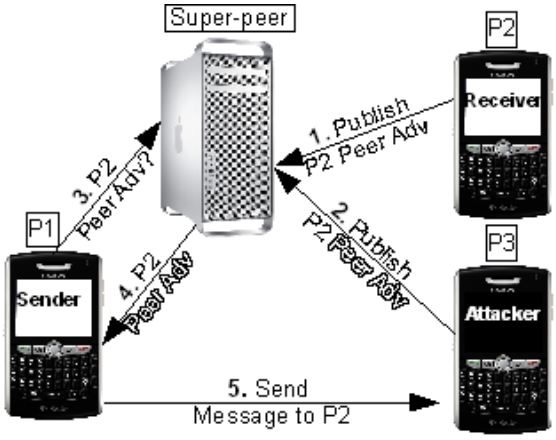

Fig. 2. Spoofing attack

malicious super-peer can perform a *Man-in-the-middle* attack between Proxyless Peers inside different networks, and modify any information in the Advertisements prior to propagating them over the network.

As far as *local data alteration* is concerned, since Proxyless Peers' local cache is stored in RAM, and no persistent copy ever exists, they are protected on the long term.

4) Message exchange: This is the most common operation in Proxyless Peers, and therefore, where more efforts have been made by developers to implement security mechanisms. Proxyless Peers exchange messages using pipes, unidirectional and virtual connections between abstract endpoints. JXME-Proxyless supports two different pipe types: Unicast and Propagate. Both are considered unreliable, however, the former is used for point-to-point communications whereas the latter is for one-to-many message broadcasts.

Unfortunately, both pipe types have some security issues:

- All data is sent in clear, and thus vulnerable to *eavesdropping*.
- Any data sent through a pipe may actually hop across other peers before reaching the intended destination, which makes the transmission prone to *man-in-the-middle* attacks.
- There is no assurance that a pipe is connected to the specified peer (*Spoofing*).

Fortunately, developers are currently working in the implementation of a wire transport level security layer that may be applied to pipes. This security layer is based in JXTA's own definition of *Transport Layer Security* (TLS) [18]. This definition is based on two distinct protocols:

- Handshake Protocol: Initial TLS negotiation protocol, responsible of the authentication between both peers.
- Record Protocol: Provides a private and reliable communication channel by encrypting data using symmetric cryptography and using integrity check.

This implementation basically allows private, mutually authenticated and reliable communication, protected against both passive and active attacks. Pipes based on TLS are named UnicastSecure, greatly improving JXME-Proxyless security.

However, after performing several test, we have realized that UnicastSecure pipes, although they provide a secure communication channel, they remain being unreliable. You can receive an acknowledge from your messenger but your message had not be sent. This is because they have been built using the *NonBlockingOutputPipe* java class. Moreover, to perform secure communications using UnicastSecure pipes, an external certificate authority (CA) responsible to manage certificates is required.

Furthermore, unfortunately, UnicastSecure pipes are restricted to point-to-point communications, and therefore cannot be used for message broadcast, which is quite common in a Peer Group context. In addition, no traffic masquerading mechanism is implemented, so it is still open to *traffic analysis*. Finally, the classes needed to implement the TLS are not inside the common libraries provided by Java ME. It requires the *Foundation Profile*, an optional package which is a standard Java specification and it is defined by the Java Community Process (JCP) in JSR 219 [19].

5) Disconnection: Since this operation does not require any communication using the network, no security vulnerabilities exist. This step in JXME-Proxyless is included just for the sake of completeness.

C. Evaluation summary

Even though no software is fully free from bugs, it can be considered that JXME-Proxyless has a big advantage because of its Open Source Software (OSS) nature [20]. Being supported by a community of enthusiastic developers, it can be considered relatively safe from *Software security flaws* on regards to security.

The analysis of possible attacks and the existing security mechanisms of JXME-Proxyless, classified by peer operations, provides a vulnerability map summarized in Table I. Attacks are those described in Section III-A, indexed by abbreviation.

From our experiments, it can be concluded that JXME-Proxyless is vulnerable to the following kinds of attacks:

- **V(1)**: Malicious executable code can easily be built and cannot be automatically discovered when installed.
- **V(2)**: No encryption mechanism exists. Advertisements are transmitted in plain text.
- **V(3)**: No data flow masquerading mechanism exits. It is easy to identify important peers by its traffic.
- **V(4)**: No repudiation method exists. Any peer can publish Advertisements in name of any peer.
- **V(5)**: No repudiation or encryption method exists. Any peer can modify Advertisements.

The available security mechanisms are:

- **P(TLS)**: Transport Layer Security

IV. Brief comparison between JXME-Proxyless and JXME-Proied

Even though we have focused on the JXME-Proxyless version, in this section we highlight the main differences with the JXME-Proxied security model. Such differences are mainly based on the fact that JXME-Proxied's main design goal is to minimize the consumption of device resources. Any peer using JXME-Proxied is named a *Proxied Peer* and since they are assumed to have very limited resources, cannot directly communicate with other peers within the JXTA network. All messages are exchanged through a *Relay Peer*, a special kind of super-peer which implements the Relay and Proxy JXTA services.

The communication between Proxied and Relay Peers is performed with a simplified protocol based on HTTP. This protocol is performed exchanging text plain messages which contain the operation to execute. The available operations are predefined: **Join** a group, **Search** or **Create** resources (such as Peer Groups or pipes), **Listen** to a pipe to receive data, **Send** data to a specific pipe, **Close** a pipe and **Poll** the Relay Peer for incoming messages from the JXTA network. Basically, it means that Proxied Peers delegate JXTA communications to the Relay Peer and only execute the previous mentioned operations.

The most important difference between both JXME versions is that a Proxied Peer needs a Relay Peer to participate in the JXTA network. Therefore, during the platform startup operation, a Proxied Peer, in addition to loading the required libraries, needs to connect to any available Relay Peer. In this initial communication, the Relay Peer creates the PeerId for the Proxied Peer and sends it in plan text. Since Relays Peers are only able to identify Proxied Peers by their PeerIds, interception becomes a security vulnerability.

In JXME-Proxied, unlike Proxyless, Proxied Peers can only join to Peer Groups which implement the *None Membership Service*, the default Membership Service in JXTA. It is designed for applications with no security concerns, being used in groups without authentication, where any peer can claim any identity. There is an initial implementation of a Membership Service based on passwords, the *Passwd Membership Service*. However, as mentioned in the JXTA documentation, it was designed only for testing, since passwords are still transmitted in clear text across the network. Therefore, we can consider that no effective security is implemented in JXME-Proxied at the join operation.

As far as Advertisement publication is concerned, in contrast to JXME-Proxyless, where they are encoded in XML, JXME-Proxied exchange plain text messages, because in limited devices a XML parser is not feasible. But in terms of security, no security layer over Advertisements is provided either. Therefore, both versions share the same vulnerabilities.

Another important difference in JXME-Proxied exists in the message exchange step. In JXME-Proxied it is only possible to perform outbound HTTP connections, in contrast with JXME-Proxyless where direct input and output TCP connections are allowed. That's one of main the reasons why Relay Peers are required. This approach tries to mitigate resource consumption, since Proxied Peers not being directly connected to the JXTA network, they do not have to forward messages, find routes or save Advertisements. However, in terms of security, while Proxyless Peers can directly send the data in

Operation/Threat	Evs	TAn	Spf	MitM	Rp	LDA	SFF
Startup	N/A	N/A	N/A	N/A	N/A	V(1)	P(OSS)
*Join**	N/A	N/A	N/A	N/A	N/A	N/A	P(OSS)
Publish/Discover	V(2)	V(3)	V(4)	V(5)	N/A	N/A	P(OSS)
Messaging	P(TLS**)	V(3)	P(TLS**)	P(TLS**)	P(TLS**)	N/A	P(OSS)
Disconnect	N/A	N/A	N/A	N/A	N/A	N/A	P(OSS)

TABLE I
JXME-PROXYLESS PEER OPERATION CYCLE SECURITY SUMMARY
(N/A: NON-APPLICABLE. V(TYPE): VULNERABILITY EXISTS. P(MECHANISM): SECURITY MECHANISM USED)
(*STEP NOT ACTUALLY IMPLEMENTED IN JXME-PROXYLESS)
(**NOT USABLE FOR MESSAGE PROPAGATION)

a secure way, Proxied Peers send it in simple text, becoming totally vulnerable to passive and active attacks.

Finally, the Disconnection operation is different in JXME-Proxied, since it is not explicit. A Relay Peer decides when to unsubscribe any Peer or a Peer Group. Proxied Peers can only perform an operation to close a pipe by knowing its id. However, it means that a Proxied Peer is vulnerable to spoofing, even when it is disconnected, until the Relay Peer decides to actually unsubscribe it.

V. CONCLUSIONS AND FURTHER WORK

Even though JXME-Proxyless is supposed to be a version conceptually very similar to desktop JXTA, with lightweight versions of the original core services, its security capabilities are still at its infancy. Only secure pipes have actually been paid attention by the developers. This is one of the evidences that being an OSS project is both boon and bane. On one hand, anyone may audit the code, looking for flaws, and contribute to the project. But on the other hand, implementing actual improvements whole depend on contributors' goodwill or interest.

From the security analysis of the JXME-Proxyless, it can be concluded that, in the current version, developers have started to take into account security, with the inclusion of TLS. Unfortunately, it is important to highlight that only using TLS is not enough to protect the system, since an attacker can easily claim any identity and impersonate any Proxyless Peer during Advertisements publication. Therefore, there's still a lot of work pending. Finally, we can also conclude that the JXME-Proxied version, were priority is in performance and not security, does not have an appropriate security baseline, because messages are exchanged with the Relay Peer in clear text, and no powerful authentication method is provided.

Further research includes providing JXME-Proxyless with an actual Membership Service, to provide authentic peer identities within a Peer Group. Once this service is established, it is possible to protect Advertisements. All these improvements should heavily take into account the idiosyncrasies of mobile devices, in contrast to a desktop environment.

ACKNOWLEDGEMENTS

This work has been supported by the Spanish Ministry of Science and Innovation, the FEDER funds under the grants TSI2007-65406-C03-03 E-AEGIS, CONSOLIDER-INGENIO CSD2007-00004 ARES.

REFERENCES

[1] Skype, "Skype on your mobile", 2004, http://www.skype.com/mobile.
[2] G. Kortuem, "Proem: a middleware platform for mobile peer-to-peer computing", *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 4, pp. 62–64, 2002.
[3] B.G. Christensen, "Experiences developing mobile p2p applications with lightpeers", *Peer-to-Peer Computing, IEEE International Conference on*, vol. 0, pp. 229–230, 2006.
[4] Sun Microsystems, "Project JXME", 2003, https://jxta-jxme.dev.java.net.
[5] Sun Microsystems, "Project JXTA", 2001, http://www.jxta.org.
[6] T. Piedrahita and E. Montoya, "Performance analysis of JXTA/JXME applications in hybrid fixed/mobile environments", *Revista Colombiana De Computación*, vol. 7, no. 1, 2006.
[7] J. Arnedo-Moreno and J. Herrera-Joancomartí, "A survey on security in JXTA applications", *Journal of Systems and Software*, vol. 82, no. 9, pp. 1513 – 1525, 2009.
[8] Sun Microsystems Inc., "JXTA v2.0 protocols specification", 2007, https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.
[9] T. Lindholm and F. Yellin, *The Java virtual machine specification Second Edition*, Sun Microsystems, 1999.
[10] Sun Microsystems Inc., "J2ME building blocks for mobile devices. white paper on KVM and the connected, limited device configuration (CLDC)", 2000, http://java.sun.com/products/cldc/wp/.
[11] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J.C. Hugly, E. Pouyoul, and B.Yeager, "Project jxta 2.0 super-peer virtual network", Tech. Rep., SunMicrosystems,Inc, May 2003.
[12] G. Paroux, I. Demeure, and D. Baruch, "A survey of middleware for mobile ad hoc networks", in *Technical Report 2007/D004*. 2007, Ecole Nationale Supérieure des Télécommunications.
[13] M. Bisignano, G. Di Modica, and O. Tomarchio, "Jmobipeer: a middleware for mobile peer-to-peer computing in manets", in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, June 2005, pp. 785–791.
[14] C. Blundo and E. De Cristofaro, "A bluetooth-based JXME infrastructure", in *Lecture Notes in Computer Science*, 2009, vol. 4803/2009, pp. 667–682.
[15] X. Wang, "Collaboration instance manager of ubicollab 2008", 2008, Master Thesis in Norwegian University of Science and Technology.
[16] D. Brookshier, D. Govoni, N. Krishnan, and J.C. Soto, *JXTA: Java P2P Programming - Chapter 8: JXTA and Security*, 2002, http://java.sun.com/developer/Books/networking/jxta.
[17] G. Combs, "Wireshark", 2006, http://www.wireshark.org/.
[18] The Internet Society, "The Transport Layer Security (TLS) Protocol Version 1.1", 2006, http://www.ietf.org/rfc/rfc4346.txt.
[19] Java Community Process, "Java specification requests (JSR) 219: Foundation profile 1.1", 2003, http://jcp.org/en/jsr/detail?id=219.
[20] J.H. Hoepman and B. Jacobs, "Increased security through open source", *Commun. ACM*, vol. 50, no. 1, pp. 79–83, 2007.