Security and Privacy Issues in Middleware for Emergency and Rescue Applications

Matija Pužar, Thomas Plagemann
Department of Informatics, University of Oslo
Oslo, Norway
{matija, plageman}@ifi.uio.no

Yves Roudier
Institut Eurécom
Sophia-Antipolis, France
yves.roudier@eurecom.fr

Abstract— **Mobile ad-hoc networks (MANETs) are a natural candidate for communication and information exchange in emergency and rescue operations. The personnel's movements, network disruptions and other system dynamics make it hard to implement robust applications for such environments. The MIDAS project aims at creating a middleware platform to simplify the task of developing and deploying mobile and robust services for events in which the network might be set-up at short notice. MANETs may be used because infrastructure is non-existing, and the number of users might be very high. One of the application domains addressed by MIDAS are emergency and rescue operations. To get a broad acceptance of the MIDAS solutions, security and privacy issues need also to be addressed. In this paper, we analyze the security threats and present a two-way approach to securing the MIDAS architecture. In the bottom-up approach, we use an efficient key management protocol to establish trust, and in the top-down approach we use dynamic role based access control to secure the system and provide privacy.**

Keywords-security, access control, mobile ad-hoc networks

I. INTRODUCTION

The MIDAS project [4] aims at developing a middleware platform to simplify and speed up the task of developing and deploying mobile services for events in which the network is set-up at short notice. MANETs may be used because infrastructure is non-existing, and the number of users might be very large. Besides large sports events like Tour de France, one of the main scenarios for applications developed using the MIDAS platform are emergency and rescue operations, where MANETs are used by the rescue personnel. Such operations are very dynamic and unpredictable, both when it comes to personnel's movements and membership, causing network disruptions and frequent changes of nodes within it. Therefore, the project has put special emphasis on the development of a data space for sharing information and a context space for managing context data as reliably as possible, to ensure the high availability of services and data, and their graceful degradation in case of long term network partitioning. While these non-functional requirements are essential for emergency and rescue applications, security and privacy also need to be addressed in order to get a broad acceptance of the project results for the application domain of emergency and rescue.

This paper presents a first conceptual study of how security and privacy can be supported within MIDAS and which security and privacy issues cannot be solved through middleware. These solutions are especially tailored for the emergency and rescue domain, but not necessarily for MIDAS only. Most of the concepts could also be applied to other distributed software of the same domain running over MANETs.

The core idea of our design is to combine bottom-up and top-down approaches. The bottom-up approach is based on earlier research results, namely the design of an efficient key management protocol, which enables us to efficiently agree on a shared key to achieve trust among the devices of emergency and rescue personnel. The top-down approach relies on the fact that roles typically define the duties and rights of rescue personnel. Due to the dynamics of rescue operations, individuals might change roles during the operation, which highlights the need for a dynamic role based access control system.

The remainder of this paper is structured as follows: Section II briefly describes the MIDAS architecture and Section III the basic assumptions for our solution. A summary of the key management protocol is given in Section IV. Section V introduces dynamic role based access control. Section VI describes how these two ingredients work together and provide security and privacy in the MIDAS middleware. Conclusions are given in Section VII.

II. THE MIDAS ARCHITECTURE

The MIDAS Architecture is shown on Fig. 1. The main components in MIDAS are *MIDAS Data Space* (MDS), *Context Space* (CXS) and *Communication and Routing* (CRT). *Access Control* (AC) *and Network Security* (NS) are new security components that are the contribution of this paper.

MDS is a relational data space provided to applications and other MIDAS components, shared between all the nodes [9]. The users and applications do not know where exactly in the network data are stored and they only see one single large data space. The MDS component does this by hiding the physical location of data from the applications, as well as by taking care of replicating data when there might be possibility for network partitioning. Replication of data to other nodes having an instance of the same table is done either eagerly (i.e. as soon as a change is done) or lazily (for nodes joining the network partition at a later point). From the application developers' point of view, only SQL queries have to be issued in order to

use the shared data space; data are published to the data space from where they can later on be retrieved.

The CXS component is responsible for defining and managing all context data in the context space. It provides middleware support for domains by implementing domain-related middleware services. As each MIDAS mobile service targets events of a specific domain (emergency operations, sports events), the support of domains helps the developers to create very focused applications within their domain. CXS allows for simple or complex (synthesized) queries and uses MDS to store and retrieve all context information. In addition, the component offers the applications services for the context addressable messaging, where message recipients are defined as a context (for instance, *"all nodes with role MEDIC that still do not have a patient assigned"*).

The CRT component is responsible for providing connectivity between nodes running the MIDAS middleware. Networks supported by CRT may be disruptive and even consist of different network technologies, which should all be seamless from the applications' and other components' point of view. CRT has mechanisms to allow connectivity even in cases when nodes are not in direct range of each other. If two network partitions have each a connection to the Internet (for instance through a GPRS connection), they can register at a pre-defined central node. As a consequence, they can merge into a single network by establishing a link between each other through the Internet.

Applications connect to the middleware using the Java Remote Method Invocation (RMI), in which the core middleware component acts as a *façade* towards its components, i.e., it internally performs calls to the requested methods in the respective components. Internally, components use each other's services either by direct calls or through the core component.

III. Basic Assumptions and Our Approach

There are five basic assumptions that were taken into account when designing security solutions for the MIDAS Architecture.

- Members of the rescue personnel (users) can be trusted.
- Devices are preconfigured before coming to the scene; this includes installation of the necessary software, certificates, etc.
- Users carry tokens with personal certificates containing their credentials (name, rank / role within the organization, etc.) and use them to authenticate themselves towards the device.
- The user-to-device authentication process is assumed to be secure enough
- Software developed on top of the MIDAS platform is considered clean (that is, it is not vulnerable, there are no viruses, trojans, etc.)

The overall security of the system depends on these assumptions to be satisfied beforehand.

The wireless medium is especially vulnerable to attacks and intrusions by malicious persons with the goal to either just

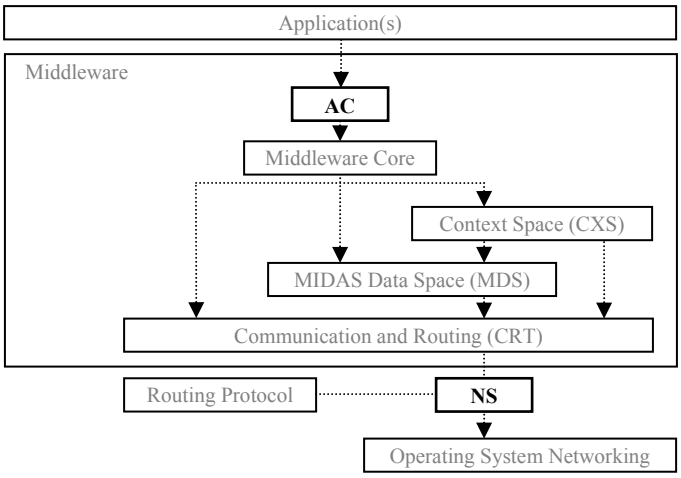

Fig. 1. A secured MIDAS architecture

disrupt the functionality of the network or to steal sensitive data, from health information about patients to possible police records. External attackers, however, are not the only issue. Not all the personnel are supposed to have access to all the data within the network. Policemen should, for example, not be able to access patients' medical records (*privacy*). Following the development of the situation, the personnel's roles and assignments might also change dynamically, providing additional challenges to the component enforcing *access control* (see Section VI for details). Since accidents naturally happen where they are least expected or they might in severe cases cause existing communication infrastructure to collapse, nodes may be left with no reliable possibility of contacting a central authority to, for instance, check a user's credentials (*authentication*). Next, it must be ensured that unauthorized nodes cannot modify existing traffic or introduce new data into the network (*integrity*). Finally, it must be ensured that receipts of given orders cannot be denied at a later point (*non-repudiation*).

We present a two-way design towards solving the abovementioned security issues, consisting of a *bottom-up* and a *top-down* approach.

The *bottom-up* approach ensures that data coming from the network are authenticated and that their content's integrity is verified already at the network layer. The whole process is hidden from the users and no intervention is required from them. This is achieved by using a key management protocol such as SKiMPy [10] to ensure that only authorized nodes are allowed to participate in forming network. Failure to do so might result in a very small percent of misbehaving nodes being able to disrupt the whole network [5].

The *top-down* approach is used to determine what data, services and other resources a certain user has access to. The access control component does this by taking into account the user's identity and role when accessing the middleware.

There are still a number of issues to be taken care of, in order for the security level to be satisfactory. Some of these issues (such as denial of service attacks on the physical layer, devices being lost or stolen, etc.) cannot be solved by the middleware and must be addressed separately. Other issues can be solved by the middleware and, as such, they will be looked

into as part of current and future work, including issuing temporary certificates, revocation of certificates for nodes being lost or stolen.

IV. The Key Management Protocol

SKiMPy is a self-organizing distributed key management protocol used at the network layer to secure either all the data coming from and to the node, or only the routing protocol (such as OLSR [1]). SKiMPy uses certificates that are installed on the rescue personnel's devices in advance of the emergency event, i.e. before coming to the rescue scene. These authorized nodes can be called *active* members of the network. All the certificates are, at the top of the hierarchy, signed by the same Certificate Authority and can thus be cross-verified by any node without the need for contacting a central authority. That way, unauthorized nodes can be excluded from participation in all or some network activities, while trust can be achieved among arbitrary devices of rescue and emergency personnel without access to the Internet. Not having contact with a central authority leads to problems in case it is necessary to revoke certificates of devices that have been lost or stolen. If anyone could issue a certificate revocation, this could also be done by the same device that is supposed to be cut out of the network, causing a network breakdown. On the other hand, a voting procedure might take place, where for example *k out of n* signatures would be necessary to issue a valid revocation. While one could argue that the authentication of the user to the device provides a certain level of security to assure that lost or stolen devices cannot be misused, this problem needs still to be solved to provide full security.

Contrary to the main exclusion policy, it is sometimes useful to grant spectators or media the ability to help at an accident scene. Such nodes might then be issued temporary certificates on the spot to be able to exchange non-critical data (i.e., become a *passive* member of the network, yet unable to influence the core network's routing tables). Such temporary certificates are then used by the same key management protocol to establish a second shared key between the passive members. The certificates would be issued by authorized users, depending on the security policies. In addition to a much shorter validity time of credential, the abovementioned authorized users would become local authorities in charge of temporary certificates' revocation with much less likelihood of losing contact.

It has to be noted that the shared key achieved by using SKiMPy can only be used to verify a message's integrity, not its source. The same applies in the case when the key is used to encrypt data. That is, it can only be shown that a message has been sent by *some* authorized node, not which one exactly.

V. Dynamic Role Based Access Control

The distributed and shared nature of the MIDAS Data Space, explained in Section II, can cause specific issues with regards to access control.

There exist several methods of implementing access control. In Discretionary Access Control (DAC [2]), users can grant or remove other users access to resources they themselves have access to (a good example for a DAC system is the UNIX file system). Due to the shared nature of MDS and the number of organizations involved in emergency and rescue operations, this method is not expressive or flexible enough. In Mandatory Access Control (MAC [2]), in contrast, resources have security labels, while users have security clearances, which then must be high enough to be able to access a given resource. Organizational hierarchy can be the starting point for defining Role Based Access Control (RBAC [3]) rules used to determine access privileges certain roles have towards certain objects. Having indeed hierarchical organizations involved in emergency and rescue scenarios, a RBAC solution, such as OrBAC [6], is a clear candidate in our case.

The users identify themselves towards the OS or towards the middleware directly, while their role or roles are, as a rule, predefined by a person's rank in the organizational hierarchy. These *a priori* roles are defined in the user's certificates. However, the users' roles can be dynamically changed during the rescue operation, depending for instance on the order in which personnel come to the scene. It should be possible to reflect these changes in the roles used for access control to data and resources. Since roles are predefined and embedded within certificates, they cannot be changed. One step towards a solution might be to use the ideas from [8], where delegation and supervision are introduced. However, additional mechanisms might be necessary, e.g. roles issued in separate temporary certificates (similar to the ones described in Section IV) that can be either used in addition to the preinstalled ones, or they might supersede them. Removing certain roles, on the other hand, is in general a non-trivial issue, and it must be ensured that a user cannot choose to ignore a newly installed certificate giving them fewer rights. However, we assume that the rescue personnel can be trusted and will therefore also not take countermeasures against those certificates.

In addition to the standard access control rules (e.g. *"can read"*), emergency and rescue operations might require a special type of rule, that is, *"must read"* and/or *"confirm receipt"*). A typical example would be an order issued by the rescue scene leader, whose receipt must be confirmed. Being signed with the recipient's certificate, this receipt cannot be repudiated at a later stage. Deontic logic introduces actions such as *"permitted"*, *"obliged"*, and *"forbidden"*. A RBAC mechanism based on deontic logic, such as the one described in [7], might be a possible solution for fulfilling both of the abovementioned needs. However, it might also be possible to implement it in a simpler way, for instance by moving some of the extra functionality towards the application.

VI. A Combined Solution

In this section, we outline how certificates and access control are used to secure the MIDAS middleware (see Fig. 1). Being just a conceptual analysis, and due to space limitations, some details on particular solutions are not specified as of yet.

At the level of network interface to the operating system, all incoming and outgoing traffic must pass the Network Security Component (NS). NS should run separately from the middleware as it has to protect not only data going to and coming from not only the middleware, but also all signaling messages of the routing protocol. Outgoing traffic is signed with the shared key obtained through key management

protocol. Incoming traffic is first checked for integrity with shared keys available at the recipient. With regards to the source's and recipient's type of membership, as described in Section IV, we have the following four cases:

1. The recipient is an active member
   a) The message is signed by an active member: all the traffic is let through
   b) The message is signed by a passive member: only non-routing messages selected according to the security policy are sent through
2. The recipient is a passive member
   a) The message is sent by an active member: all the traffic gets through
   b) The message is sent by a passive member: all the traffic gets through

In addition, the incoming traffic must be checked for replay-attacks, i.e. it must be ensured that a valid message cannot, at a later point, be re-sent by a malicious node. For instance, routing messages that are re-sent at a later point may cause the whole network to collapse.

The middleware's application programming interface (API) is designed so that all calls issued from the applications towards the middleware are passed through the Access Control (AC) component. It must be noted that all the inter-component calls must go through AC as well, since it could happen that a component tries to perform an illegal action. It is the task of AC to verify whether the given user has the necessary credentials for that call. In case it is a query towards the data space, the user must have the necessary rights to perform that query on the specified table or tables in the data space.

Remote queries have to be verified twice, first at the source node and then again at the destination node. This means that the issuer's credentials (including their public key) must be included in the message. The issuer's public key may be used to encrypt the response to the query, but a more light-weight solution might be preferred if such queries are expected to be performed often. The underlying shared data space could be used to distribute the credentials of all the users present at the scene, leaving up to MDS the task of keeping the information in the data space as consistent as possible.

Access rights are defined by the personnel's identities or roles, both of which are specified in their pre-installed certificates. To add or change roles dynamically, as a consequence of the rescue operation's development, personnel can be issued temporary certificates on scene. Given the possible severity of the situation, the notion of access rights should include access duties as well, meaning that some messages *must* be read by the personnel they are addressed to. That way, the receipt of an order cannot be denied at a later point, making it known who had been in charge of what at any time in the operation.

VII. Conclusion

Reliability and availability are important non-functional properties for emergency and rescue applications. The MIDAS middleware provides these properties through optimistic replication. In order to achieve also security and privacy for MIDAS in emergency and rescue applications, we propose in this paper a combined bottom-up and top-down approach. The dynamic role based access control we propose comes very close to the way security levels and rights assigned to emergency and rescue personnel are assigned in the different organizations. Access control is enforced with the help of certificates. In addition to the misuse of middleware service, it is necessary to protect the network itself. Therefore, all traffic is screened at each node and all packets from non-trusted nodes are simply discarded to disable attacks on the network.

While this approach provides a high level of security for the network and the middleware and enables to protect data, there are still certain attacks that cannot be handled, especially attacks at the link layer, like noise etc.

Our ongoing and future work is concerned with two aspects. First, we aim to refine the proposed solution and prototype it to be able to perform evaluation testing of the presented solution. Next, we extend the solution also for other application domains, like large sports events, where existing infrastructure is an important part of the network.

Acknowledgments

This work has been performed in the context of the MIDAS project (funded by the European Commission's 6th Framework Program, Contract no. 027055), and the Ad-Hoc InfoWare project (funded by Norwegian Research Council IKT2010, Project No. 152929/431).

References

[1] Clausen, T., Jacquet P., "Optimized Link State Routing Protocol (OLSR)", RFC 3626, October 2003

[2] Department of Defense (DoD) Trusted Computer System Evaluation Criteria (TCSEC) (DoD 5200.28-STD 1985), Fort Meade, MD, Department of Defense, 1985

[3] Ferraiolo, D.F., Kuhn, D.R., "Role-Based Access Control", Proceedings of the 15th National Computer Society Conference, National Institute of Standards and Technology, Gaithersburg, MD, October 1992, 554-563.

[4] Gorman, J., "The MIDAS Project: interworking and data sharing", Interworking 2006, Santiago, Chile, January 2007

[5] Hollick, M., Schmitt, J., Seipl, C., Steinmetz, R., "On the Effect of Node Misbehavior inAd Hoc Networks", Proceedings of IEEE International Conference on Communications, ICC'04, Paris, France, volume 6, pages 3759-3763. IEEE, June 2004

[6] Kalam, A. et al, "Organization Based Access Control", 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003), Como, Italy, June 4-6, 2003.

[7] Kołaczek, G., "Application of deontic logic in role-based access control", International Journal of Appllied Mathematics and Computer Science, vol. 12, no. 2, 2002, p. 269-275

[8] Moffett, J.D., Lupu, E.C., "The Uses of Role Hierarchies in Access Control", Proceedings of the fourth ACM workshop on Role-based access control (RBAC '99), Fairfax, Virginia, 1999, p. 153-160

[9] Plagemann, T. et al, "A Data Sharing Facility for Mobile Ad-Hoc Emergency and Rescue Applications", The First International Workshop on Specialized Ad Hoc Networks and Systems (SAHNS 2007), Toronto, Canada, June 2007

[10] Pužar, M., Andersson, J., Plagemann, T., Roudier, Y., "SKiMPy: A Simple Key Management Protocol for MANETs in Emergency and Rescue Operations", Proceedings of the 2nd European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS 2005), Springer-Verlag, LNCS 3813, Visegrad, Hungary, July 2005