ADAM: AN AGENT-BASED MIDDLEWARE ARCHITECTURE FOR DISTRIBUTED ACCESS CONTROL

Alexandr Seleznyov Mohamed O. Ahmed Stephen Hailes
Department of Computer Science, University College London
London WC1E 6BT, UK
{A.Seleznyov, M.Ahmed, S.Hailes}@cs.ucl.ac.uk

Abstract

This paper outlines a conceptual architecture for an autonomic middleware component designed to provide application-independent access control for use in large-scale highly-dynamic computing environments. In such environments, most notably ambient/pervasive computing environments, centralised access control policy determination is impossible or inadvisable because of the complexity of trust relationships. In the absence of centralisation, network resources are forced to make trusting decisions locally, in the light of information that they themselves can gather. Thus the architecture that is described in this paper is founded around an automatic knowledge acquisition and processing mechanism, acting as the foundations of a semi-autonomous multi-agent system (MAS). The agents dynamically organise themselves into cooperating distributed communities that mediate between users and devices (collectively known as *trustees*) and network resources (*principals*). Once activated by their owners, agents maintain user credentials, negotiate amongst themselves to establish the credibility of prospective trustees identities and cooperate to gather evidence about the likelihood of trustees adhering to the policies of principals.

Key Words

Trust, authorisation, access control, ambient, autonomic

1. Introduction

Our society has become increasingly dependent on the rapid access to information and services. The increasingly good connectivity demanded by users enables access to larger, more varied and more widespread resources more quickly than ever before. However, not all such access is benign; the number of security incidents on conventional networks has grown significantly over the past few years. Unfortunately, from this point of view, the networked world is becoming ever more dynamic; the application mix is constantly changing and, with the projected development of ambient systems, connectivity will change even more rapidly than for mobile systems, and organisational boundaries will become less clear. Taken together with the failure of PKI as a manageable technology, and the difficulties in providing manageable intrusion detection, the uncertainties in this environment mean that it is extremely difficult to establish trust relationships between network entities.

This work is intended to address the problem of making access control decisions in a distributed fashion in the absence of organisationally imposed policy constraints. Such decisions can be considered a decision by a network resource (principal) to trust a user or device (a trustee). The ambient computing vision is one that particularly benefits from this approach; in this, there are many devices and many services, controlled by many organisations, with rather unpredictable interactions between them. The number of possible bilateral trust relationships can become very large, and without the imposition of a hierarchical organisational structure or some other means of stereotyping, a priori definition of trust relationships becomes impracticable.

In order to facilitate the operation of an ambient computing system, and to ensure that an appropriate and timely response can be made to requests for authorisation, it is necessary that the architectural components of a trust management system have a degree of autonomy. Unfortunately, except in very highly controlled single-domain environments, this means that the concepts of equality and certainty in trust must be abandoned.

An attempt to design an abstract, comprehensive framework to analyse and manage trust relationships was made by Grandison [1], resulting in the Simple Universal Logic-oriented Trust Analysis Notation (SULTAN). SULTAN was developed to simplify the specification of trust relationships and their management. Again, although a generic specification was proposed, there is no efficient algorithm to automatically perform trust management operations. SULTAN uses a goal-orientated planner to refine rules, which often require a network of policies (both authorisation and obligation), rendering the approach unscalable in complex situations, because the refinement process can become intractable. Additionally, although the notions were defined, there is no built-in mechanism supporting trust recommendations (referring) rules.

A mathematical framework to evaluate trust management systems was proposed by Weeks [2]. The framework is designed to allow expression of trust management systems making it easier to understand, compare, and design rules. In this work, the semantics of a trust management engine were defined, which lead to an efficient implementation in some situations. The proposed framework is insufficiently generic to be applicable to environments such as ambient computing environments. Further, the framework is only targeted at monotonic systems.

Although the trust management problem has been extensively discussed, most currently proposed trust management systems make their trust decisions in a largely ad hoc and inflexible way [3]. In particular, these solutions do not take into account concept drift, assuming that trust relations do not change over time. Consequently, their trust models are oversimplified because local trust decisions are based on inhomogeneous shared trust resources that are constantly changing. Having no control over these changes, a trust management system must adapt its behaviour to suit the environment; failure to do this significantly increases its error probability in the future. By hard-coding trust decisions into trust management systems, developers make systems inflexible, denying them the ability to learn from past experience and adapt to changes in the environment by modifying their behaviour. Consequently, a generic, autonomous trust management system must deal efficiently with inconsistencies in trust relationships arising from differences in security policy definition and management [4].

In this paper we propose a holistic approach to building a resilient, dependable and fault-tolerant trust-management system to manage access control to network resources. Placed between network and application layers, it thus forms an application-independent middleware component, enabling us to create a scalable and flexible approach for managing distributed environments, and providing the possibility of handling issues such as concept drift and uncertainty through observation of and changes to the operational environment. Distributed knowledge acquisition and management is used to authenticate users and aid in reasoning about their credibility when establishing appropriate trust levels authorising (or declining) access to requested resources. We use a distributed multi-agent architecture for a flexible, general-purpose management of the security of resources in networks. The distributed architecture allows the system to query different information sources dispersed over a network (or networks) to build comprehensive knowledge about users. In view of its self-organising and self-protecting nature, it is, by definition, autonomic.

The reminder of this paper is organised as follows: Section 2 defines the notion of trust discussing attributes of trust relationships. In Section 3 we discuss conceptual architecture of Autonomic Distributed Authorisation Middleware (ADAM). Section 4 outlines the process of distributed authorisation. Finally, Section 5 concludes this paper with final remarks.

2. Premises

To date, many definitions of trust have been proposed, and we will not present a new definition in this work, but instead adapt an existing definition and concentrate on the management of trust relationships. The focus of this work is on building an efficient system that controls the full life-cycle of a trusting relationship, starting from its establishment thought to its revocation. The proposed system incorporates reactive elements that monitor the manipulation of network resources and respond when malicious activity is detected. However, these elements are beyond the scope of this paper. For our work, we adapt the definition of trust in a way that treats it as *a measure of willingness of a responder to satisfy an inquiry of a requestor for an action that may place all involved parties at risk of harm, and is based on an assessment of the risks and reputations associated with the parties involved in a given transaction*.

As the above definition states, all parties involved in interaction may be harmed through its consequences. By providing access to its resources, a principal may be explicitly harmed by the malicious actions of trustees. Third parties involved with the principal in existing trust relationships may be implicitly compromised by the relationship. Trustees are at risk, since obtaining incorrect or contradictory information compromises the integrity of their knowledge bases and those of their trusting parties. These situations may arise as the result of intentional malicious activity or through legitimate error such as software bugs, network failures, and user errors.

The potential risk of undesirable outcomes is assessed by checking the credentials of participants and the history of their behaviour with respect to the value of the resource they request; only then can a decision be made about whether the risks posed through interaction are acceptable. To support this, we employ a dynamic mechanism for object manipulation tracking, to perform accounting and the detection of unusual patterns of behaviour or resource usage.

In establishing trust relationships, the following attributes are important: the *participants*, the *scope* (spatial and temporal restrictions applied to a current trust relationship) of the relation, the *risk* (as a degree of potential damage) associated with the relation, and *security* to describe the characteristics of trust. Each access control relation expresses: sets of *principals* that use and provide *resources* in the environment, *actions* that may be applied to the resources, and *policies* governing the use of the resources.

An interaction starts with a request for a service; requests in ADAM are tuples, consisting of a *resource*, the set of *actions* to be performed and the *identity* of the

user. In order to facilitate this, there must be a service/resource discovery mechanism that allows prospective trustees to look for services, view the actions available and the credential requirements of the services and choose between them. ADAM does not itself perform authentication, it only authorises. When a user submits her request, an authorisation-agent is launched to collect evidence about the behavioural disposition of the identity presented. There are three main sources of information from which entities' trustworthiness can be derived [5]: The first source is from *direct observations* that are formed by recording outcomes of previous interactions with an entity. The second is the *recommendations* of trusted entities that allow the propagation of trust. The third source is *reputations*. Reputations are knowledge about an entity's behaviour derived from their history of interactions. They may be derived form the common knowledge of an organisation or a community or based on specific pre-defined knowledge [6]. For our system we use knowledge that is accumulated over time in communities of practice to assess reputations of network entities.

During the authorisation procedure, the authorisation-agent also performs risk assessment and checks whether the potential risk posed by the interaction is acceptable in terms of the local policy of the resource. It assesses the potential damage to the user's reputation and the potential for identity loss or loss of associated values (money may be charged if a credit card number is provided). It analyses this alongside the potential damage the resource may incur. Thus it may be the case that a user will be declined access when using one of her electronic identities and granted access when using another.

To summarise this section, we would like to emphasise several foundational aspects of ADAM that make it different from the other trust management systems:

- It is designed to work in different networks allowing automatic trust establishment and maintenance between entities situated in different network and administrative domains. This provides additional flexibility and allows ADAM to function in a ambient computing environment.
- It allows each user to have multiple electronic identities.
- It only authorises, it does not authenticate. The authentication task is delegated to separate parties.
- ADAM authorises transactions (actions), not users.

In the following sections we will discuss ADAM in detail, giving some practical considerations relating to its implementation.

3 System Architecture

Authorisation decisions in ADAM are the product of negotiations between two agents: *user agents* and *authorisation agents*. The former represent users interests in negotiations and are implemented as mobile agents, aware of local policy on the user side. On the other hand, the authorisation agents are not mobile and are aware of the policies and represent the interests of the resource providers.

3.1 User Agents

The user-agent contains information about its legal owner, including secret keys, and certificates[1] required for the user authentication. Within a user-agent, all information is encrypted. To activate a user-agent a correct PIN or password is required; PINs constitute parts of decryption keys for user agents. Users may not access user agents whilst they are inactive or in transit. The mobility feature allows agents to move between networks or devices; for example, for the user's convenience, an agent may be a resident on their PDA. Having user agents not only simplifies users' workload, they make network management easier because they can be used to automate certain tasks, such as password and certificate management, the collection of relevance feedback information etc.

Clearly, for reasons of security, user agents must perform all activities involving private keys. Whilst it is inherent in the model that there is the possibility that malicious users can get to information from an agent, we consider the risk to be several orders of magnitude smaller than the risk of discovery of some or all the PINs used to activate an agent (see [7] for further analysis of PIN security in different networks). It has been argued that, from a technical perspective, this kind of user information storage does not bring new security threats to those to already present in computer networks [8].

3.1 Authorisation Agents

Authorisation agents protect network resources by ensuring that only valid users can obtain access to them. They are aware of and enforce the policies and procedures of the resources they manage. In conjunction with an intrusion detection module, they can apply real-time access control restrictions to enable reactive security measures.

3.2 Information flow in ADAM

An interaction involves two interested parties that are potentially willing to cooperate: a client searching for a resource, and a service willing to provide the resource. The goal of the interaction is to reach a mutually beneficial outcome.

[1] It may include some other information, such as credit cards numbers, user names and passwords for different resources, etc.

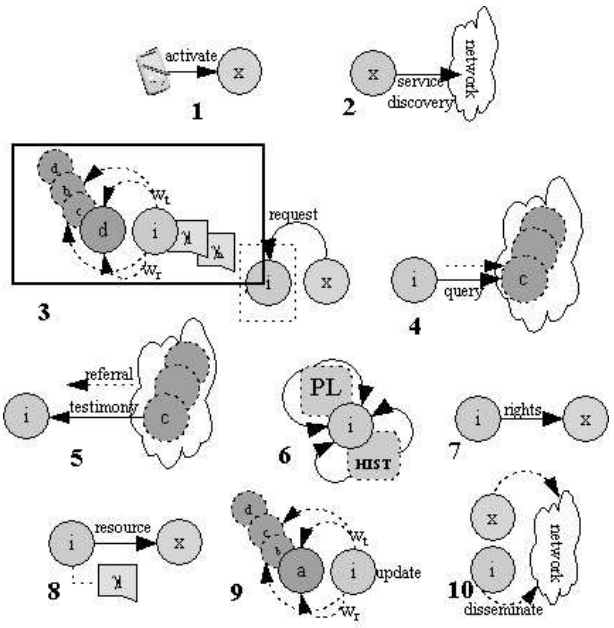

Figure 1 Information flow in ADAM

As shown in Figure 1, the process starts with the user-agent activation (1) as discussed in Section 3.1. Once activated, the user-agent (*x*) performs service discovery, during which information about different services advertised in the network is gathered (2), this information includes descriptions of services available, permissible actions, and information about the credentials required to obtain resources from each service – as described in the local policy of each service provider.

It is at this point the user chooses the most suitable service, being aware of the required credentials. Clients may wish to evaluate their chosen service before using it. This evaluation is performed with help of local policy, and may necessitate the gathering of additional information about the quality of the service by collecting opinions of other clients who have previously used it. If, up to this point, the client is satisfied, they make a request to the service provider (3) to use a resource (γ). Requests are handled by authorisation agents (*i*) using the local policy of the service (*PL*). We place no strict constraints on the criteria by which requests may be evaluated, to allow service providers the flexibility of choosing the criteria they value as important.

When clients provide the credentials required to use a service, the authorisation-agent collects two types of evidence (4). First is evidence that the credentials are correct and that the requesting user is indeed the legitimate owner of those credentials; this information is used to authenticate the user. Second is evidence about the behavioural disposition of the identity, expressed through its reputation in the environment and collected as testimonies from other network entities. Authorisation agents do not perform authentication, instead authentication is delegated to third parties that have had previous experience of interactions with the user or different authorities. To do this, an automatic credentials discovery (ACD) protocol is used (to be discussed in a forthcoming coming paper).

There are various sources that can aid the verification of users' credentials and provide feedback about users' reputation (5). However, in an ambient computing environment, it is expected that these will be distributed over numerous networks and administrative domains. As a result, there is no guarantee that they will act cooperatively and every chance that they will have local policies that limit the amount and nature of information they are allowed to share. Our system is designed to be aware of this and associates confidence weights with the testimonies (*wt*) and referrals (*wr*) of witnesses to represent the uncertainty in their information.

When information is collected, it is necessary to combine it with any history (*HIST*) of interaction with the user that may exist and transform this heterogeneous set of opinions into homogeneous data that can be automatically processed.

The result of negotiations between agents is not binary. Negotiations themselves are regulated by a set of fuzzy rules that are dynamically created and reflect local policies (6). Based on this data, the decision about user reputation is made and rights are bestowed (7). Once a user's credentials have been collected and evaluated, the authorisation agent may decide that the user's credentials are inadequate to authorise the requested action (for example, "write"), but they are enough to allow some less security sensitive actions (like "read"). The client may accept or decline the offer, or renegotiate using a different set of credentials. In the case of a positive decision, the authorisation agent creates a link associating the client with the resource (8). By controlling and maintaining this link, the authorisation agent enforces access restrictions. During the lifespan of links authorisation agents perform auditing to assess users' cooperativeness in adhering to the policies of the transaction. The audit trails are used for reactive fraud detection and response.

Authorisation agents are used also to perform anomaly detection, by looking for certain patterns of user behaviour during a transaction. In the case of misuse, the distributed architecture may be efficiently used to notify interested parties such as the authenticating parties about the problems encountered. At the end of session, when the authorised action has been completed, the link is destroyed and the authorisation agent classifies its experience as positive or negative, stores the experience, and disseminate it (10). The authorisation agent also updates its local view of the performance of the testimony providers, based on the results of the interaction (9).

To work in the real world, systems such as ADAM must be designed with uncertainty in mind. There are three sources of uncertainty for ADAM to deal with. Firstly, the credibility of credentials can easily be questioned, and so ADAM gives service providers the flexibility to request different types of identity for different classes of services. Secondly, the credibility of the testimonies regarding the authenticity of an identity or the reputation of individuals can be also be questioned,

and thus ADAM assesses the performance of third parties in fulfilling the requests it generates, thereby allowing it to learn from experience and select those it queries for information in an informed way. Lastly, the actions ADAM perceives while interacting with users may not always be interpreted correctly, and this is dealt with in three ways: (i) the separation of actions and rights ensures that the user's view of resources is in solely terms of the rights they possess (ii) collaborative filtering in the form of reputations provides some information regarding what to expect from users (iii) real-time anomaly detection and reaction is applied to users' interactions with the resource by the authorisation agent; this allows control over the effect of the actions of users. Integrity checks are aimed at exposing different problems in the system, such as intrusions, deception attempts and internal failures. They aim to allow the system to continue functioning properly in case of problems, through segregating subverted or malfunctioning agents. These components will all be discussed in detail in forthcoming papers.

Below, we discuss some key points of the decision-making process giving suggestion on their practical implementation or further research.

4. Distributed Authorisation

Decentralised trust management provides a scalable and flexible alternative to centralised systems management. The inherent decentralisation of individuals relying on trust rather obligation in sanctioning transactions represents a move away from the binary certainties of traditional methods such as PKI. Likewise, the reduction of the granularity of trust management to the service level means that policies are finer grain and can be treated as behaviours and dependencies to be dealt with using a behavioural approach. ADAM uses a subsumption-based approach to handling dependencies.

In creating decentralised, cross-domain, trust management we face a number of obstacles discussed below.

4.1 Evaluating Identities

In this work, we take a pragmatic approach and agree with Friedman et al. [3] that identity as manifested by pseudonyms is cheap and effective, whereas centralised maintenance is unfeasible in large systems without a heavy cost to the dynamism of systems. Because of this we enable systems to associate a level of confidence with different types of identities. This enables systems to associate differing levels of uncertainties with the credibility of the identity, based on whether they have existing contracts with the identity provider or have been configured to prioritise various identities over each another. Learning the reliability of different classes of identities was considered, but rejected because the value of an identity is different to every owner. For example, a web-mail identity may be abused by some users whilst being legitimately maintained by others.

4.2 Identifying Neighbours

Without the existence of global TTPs, principals must identify trustworthy associates from which to collect evidence, when judging the credibility of authentication tokens and the likelihood of trustees in adhering to the policies of principals. The main point raised by this is the question of from where principals should get information from and what value should they place on it. Principals relying on the testimonies of witnesses must act in such a way as to retain good links with those whom they perceive as providing good testimonies and referrals. This is a self-organisation problem where principals must continuously asses the performance of their neighbours, weigh their testimonies based on the accuracy of their information, and proactively manage the membership of their neighbour set. In considering whether to change neighbours, principals utilise a dynamic threshold for assessing the performance of prospective neighbours with respect to the current set of neighbours. If the threshold is breached, a principal replaces its worst performing neighbours with new ones – chosen either randomly or on the basis of perceived utility. To deal with the uncertainty in testimonies of witnesses, we associate two weights with each neighbour representing their competence in providing testimonies about trustees and referrals to other witnesses that provide testimonies. These weights are updated on the basis of the discrepancy between witness testimonies for trustees and the perceived level of cooperation from the trustees.

4.2 Contextualisation

The next problem that such systems face is in combining testimonies that may have given in or for different contexts. Context is omnipresent in the concept of trust; in ambient computing systems it is necessarily local and ontological and cannot therefore be fully addressed through clustering-based techniques. Within ADAM, contexts are expressed in queries and fall into two broad categories: *communicative* and *transactional*. Communicative context represents the social aspects of interaction between agents and takes the form of *testimonies* and *referrals*. Transactional contexts determine the local aspects of interaction, and these may be classified under *supply* and *demand,* relating to principals and trustees respectively. They specify the resources principals offer and those that trustees want to utilise. Thus principals and trustees are explicitly aware of interacting in different contexts. Combined with the assumption of uniform service description, we limit the need for context lifting when assessing information and make it easier to define context specific policy.

4.3 Translating evidence to policy

Evidence collected will eventually have to be translated into decisions that can be used to assign rights. To do this, we define the security aspect of policies in terms of the risks that actions permitted by the system pose to its *confidentiality*, *integrity* and *availability* (as proposed by

the FIPS 199 categorisation [9]). The impact of each action upon each criterion is represented as a fuzzy set, as is the level of confidence required in evidence to permit the action. Fuzzy sets are used to represent the uncertainties in risk assessment because of the difficulty in gathering representative quantitative data. The union of these sets is then used determine the risk trustees pose.

4.2 Dealing with anomalies

In ADAM, anomalies are dealt with at two levels: first, real-time behavioural tracking is used to detect anomalous behavioural patterns and identify misuse of resources. Secondly, information sharing is employed, through the implicit social cohesion enabled by self-organisation. Taken together, this provides us with two useful properties: (i) dense information channels that keep principals up to date about the behaviour of trustees without centralised maintenance (ii) an implicit enforcement mechanism because principals will rate the opinions of those that they perceive as providing good testimonies and referrals, thereby creating exogenous incentives for trustees to act cooperatively in the environment, so as not to develop a bad reputation.

5. Acknowledgements

We would like to thank the EPSRC and BTExact for their generous grants and ongoing support for this work.

6. Conclusion

This paper presents a conceptual description of the Autonomic Distributed Authorisation Middleware (ADAM), which is aimed at automation of trust establishment and maintenance process by performing distributed knowledge acquisition and management. The architecture is based upon two groups of agents: mobile user agents protecting user interests and authorisation agents protecting network resources. The access control decisions are results of negotiations between them. Local policies are translated into sets of fuzzy rules and the negotiations aimed at finding consensus between these sets. The system allows the automation of trust establishment by gathering information about network entities and later maintenance of trust by constantly controlling information flow and manipulation. Finally, it allows users to use multiple electronic identities to preserve their privacy.

ADAM is an autonomic middleware component that is independent of the type, topology and administrative structure of both networks and applications. Consequently, it facilitates automatic trust establishment and maintenance between entities situated in different network domains, which provides additional flexibility and allows ADAM to function in ambient and pervasive environments.

References

[1] T. Grandison. *Trust specification and analysis for Internet applications*. PhD thesis, Department of Computing, Imperial College of Science Technology and Medicine, London (UK), 2001.

[2] S. Weeks. Understanding trust management systems. *In IEEE Symposium on Security and Privacy*, Oakland, California, 2001, 94-105.

[3] E. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2), 2001, 173-199.

[4] Khare, R., Rifkin, A. Trust management on the World Wide Web. *Peer-reviewed Journal on the Internet*, 3(6), 1998. http://www.firstmonday.dk/issues/issue3_6/khare/index.html, [Read 09.05.2003].

[5] C. English, W. Wagealla, P. Nixon, S.Terzis, A. McGettrick, and H. Lowe. Trusting collaboration in global computing. *In First International Conference on Trust Management*, volume 2692 of LNCS, 2003, 136-149.

[6] L. Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, MIT, Cambridge, Massachusetts, December 2002.

[7] J. Tang, V. Y. Terziyan, and J. Veijalainen. Distributed pin verification scheme for improving security of mobile devices. *ACM MONET, Special Issue on Security in Mobile Computing Environments*, 8(2), 2003, 159-175.

[8] J. Veijalainen, A. Seleznyov, and O. Mazhelis. Security and Privacy of the PTP, In Makki, K, Pissinou, N, Makki, K., Park, E. (Eds.), *Mobile and Wireless Internet: Protocols, Algorithms, and Systems* (Kluwer Publishers, Boston (US), 2003), 165-190.

[9] National Institute of Standards and Technology (NIST). *Standards for security categorization of federal information and information systems.* http://csrc.nist.gov/publications/drafts/draft-fips-pub-199.pdf, September 2003.