

An Access Control Model Based on Distributed Knowledge Management

Alexandr Seleznyov

Stephen Hailes

Department of Computer Science, University College London

Gower Street, London WC1E 6BT, UK

{A.Seleznyov, S.Hailes}@cs.ucl.ac.uk

Abstract

The conceptual architecture of the access control system described here is based on automatic distributed acquisition and processing of knowledge about users and devices in computer networks. It uses autonomous agents for distributed knowledge management. Agents grouped into distributed communities act as mediators between users/devices and network resources. Communicating with each other, they make decisions about whether a certain user or device can be given access to a requested resource. In other words, agents in our system perform user/device authentication, authorisation, and maintenance of user credentials.

1. Introduction

Many approaches to access control and trust management have been developed and implemented over the years. Initially, the approach taken meant that each assertion from the space of trust metrics was evaluated independently – the so-called scalar assumption [1], [2]. Levien [3] extended the notion of trust metrics by introducing the concept of group trust metrics. He claimed that group trust metrics have significantly better attack resistance properties than trust metrics formed under the scalar assumption. However, much research and development work remains in order to make such proposals a reality.

Although the trust management problem has been extensively discussed and many definitions have been proposed [4, 5, 6], it has not been properly addressed by contemporary solutions [3]. Firstly, most current solutions rely on ad hoc and inflexible mechanisms to reach trust decisions. Secondly, they do not take into account concept drift, and assume that trust relations do not change over time, an oversimplification for any environment deployed over the long term.

Local trust decisions are based on shared trust resources. Even in relatively homogeneous environments, the precise semantics attached to given statements about trustworthiness will gradually drift apart in different administrative domains in the light of innovation or change, whether in technology, applications or

management practices. Having no control over these changes, a trust management system must recognise this ontological uncertainty in a first class way, thus allowing for adaptation over time. This is a radical departure from existing approaches, in which trust decisions are hard-coded, and explicitly recognises the need to move away from simple, unachievable, certainties to best-effort flexibility in which systems automatically adapt to changes in their domain of operation and modify their behaviour accordingly.

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 remainder of the paper organised as follows: Section 2 defines the notion of trust, discussing attributes of trust relationships. In Section 3 we outline the conceptual architecture of our Autonomic Distributed Authorisation Middleware (ADAM). Finally, Section 4 concludes this paper with final remarks.

2. Premises

Many definitions of trust have been given to date and we do not intend to add to this list; instead we concentrate on the management of trust relationships by building an

efficient system that controls the entire lifecycle of a trust relationship, from its establishment to its revocation.

We adapt a definition of trust in a way that treats trust as *a measure of the willingness of a responder to satisfy an inquiry of a requestor for an action that may place all involved parties at risk of harm. This measure is based on an assessment of the risks and reputations associated with the parties involved in a given transaction.* As can be seen from the definition, all parties involved in a transaction may be harmed as a result of the transaction. A resource may be harmed since a requestor may perform some malicious activities within the given access. Third parties that are connected with the responder by trust relationships that have already been established may be implicitly affected as well. Finally, the principal is also at risk since obtaining incorrect or contradictory information compromises the integrity of its knowledge base. All the above situations might be the result of intentional malicious actions or by mistakes: software bugs, network failures, and user errors.

As a consequence of these risks, it is imperative that participants undertake a process of risk assessment before permitting a transaction to proceed. Likewise, it is most sensible to employ dynamic mechanisms for object manipulation tracking, which perform accounting and detect unusual patterns of behaviour or resource usage.

The risk assessment process involves the collection and collation of information about behaviour. There are three main sources that are usually used to collect relevant information to assess an entity's trustworthiness: (i) *direct observations*, which are formed by recording outcomes of previous interactions. (ii) *recommendations* from trusted entities that provide the possibility for trust even regarding unknown/unseen entities to be propagated. (iii) *reputation*, which is knowledge about an entity's behaviour derived from the history of its previous behaviour and/or past transactions. The process of forming reputation is lossy, in the sense that information from different sources is analysed, combined, and summarised in building a reputation. Most importantly, the binding between the identities of recommenders and their recommendations is lost.

The whole purpose of gathering the above information is to allow the establishment of a *trust relationship*, the attributes of which we define to be: participants, scope (spatial and temporal restrictions applied to a current trust relationship), risk (as a degree of potential damage), and security. The first three properties are related to the trust relationship itself, describing its features. The last – security – is related to environment in which the relationship exists. We combine all of these properties to allow our system to successfully create and manage trust relationships in different domains and networks whilst satisfying all the security restrictions of these domains.

An inevitable observation for systems of this nature is that the linkage of identity to reputation is a fundamental precondition for the remainder of the system to function correctly. We believe that this observation is actually incorrect; ADAM does indeed support systems that wish either to have strong guarantees on identity or to support pseudonymity through, for example, zero knowledge proofs. However, it also has the flexibility to allow users to create and use multiple electronic (pseudonymous) identities and thus to disassociate themselves from previous, bad, reputation and to start afresh. However, there is no requirement on resources to accept such weak forms of identification as authenticating users; the approach we have adopted merely facilitates this if the resources are happy to permit it – *caveat emptor*. Thus, ADAM is concerned with the *authorisation* of actions that users wish to perform on resources, not the *authentication* of users themselves.

To conclude this section, we would like to highlight aspects of the foundations of ADAM that make it different from other trust management systems: (i) it is designed to allow automatic trust establishment and maintenance between entities situated in different network domains, which provides additional flexibility and allows ADAM to function in a pervasive and ubiquitous environment; (ii) it only authorises, it does not authenticate; (iii) the authentication task is delegated in such a way as to permit resource-specific authentication, which encompasses everything between strong authentication and anonymity; (iv) ADAM authorises transactions (actions), not users.

3. System Architecture

In previous sections, we described the principles on which our system is founded. Here we discuss ADAM's conceptual architecture in more detail, giving practical considerations about its implementation.

Authorisation decisions in ADAM are produced as results of negotiations between two agents: user agents and authorisation agents. The former are implemented as mobile agents. They are aware of local policy on the user side and represent user interests in the negotiations. The user agent contains information about its legal user, secret keys, and certificates required for the user authentication (it may include some other information, such as credit cards numbers, user names and passwords for different resources, etc). All information is encrypted and, to be activated, a user agent requires a correct PIN or password to be entered. PINs constitute parts of decryption keys for user agents, using which user agents can only be activated by authorised people. It also denies access to user agents when they are inactive or travelling. The mobility of agents allows them to move across networks or between devices. For, example, for the user's convenience, an agent may be resident in the user's PDA.

The user agents not only simplify users' lives, they make network management easier since they automate certain tasks, such as password and certificate management. For example, to re-issue a user password, the user agent is notified. After this, it moves to the network server responsible for managing users' profiles, where the password is changed in a secure environment, making it unnecessary to transmit sensitive information over the network. This method has also other benefits. Since the user does not need to remember her password, it is possible to choose strong passwords automatically without requiring any extra activity from users.

Some information is stored in user agents for users' comfort. It is necessary to remember only one PIN or password. Once the user agent is activated it can submit some user information on request: for example, user names and passwords for other resources, certificates, etc. However, clearly, some information, such as private keys, should never leave user agents. There is still a small probability that a malicious person manages to get information from an agent. We consider this to be rather smaller than the risk of finding out some or all of the PINs used to activate an agent. In [7], it was argued that, when correctly implemented, this kind of user information storage does not bring new security risks to those already present in computer networks.

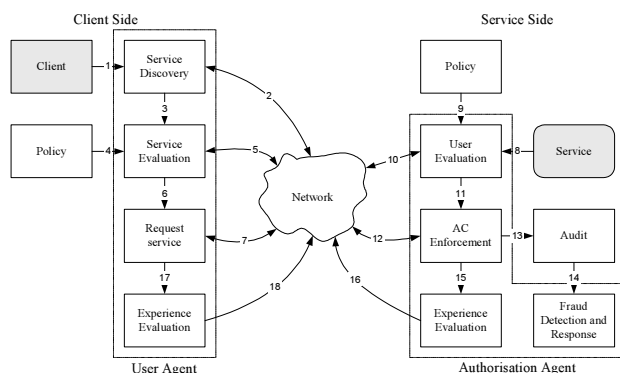


Figure 1 Authorisation process in ADAM

Authorisation agents are meant to protect network resources by ensuring that only valid users obtain access to them. Agents are aware of local policy on the resource side and enforce policy rules and procedures. Also, after the authorisation procedure, agents enforce access control restrictions and monitor usage of resources in support of reactive security.

Consider the negotiation process in detail. Figure 1 shows the main phases through which ADAM has to go to process each request. Initially, there are two interested parties that are potentially willing to cooperate: client and service. The former is looking for a service or resource to use for her needs. The latter is willing to provide this service. Firstly, the client needs to locate an appropriate server. Secondly, in order to cooperate, they must

convince each other that they are sufficiently trustworthy to perform this transaction. These actions take place in a number of steps:

Each user has to activate her user agent by giving the correct PIN/password (1), as discussed above. A secure channel is established between the user terminal and the agent. If no agent with this user's information is found, a new agent is created and assigned the task of carrying out user requests. After this, in order to find an appropriate service, the client needs to perform *service discovery*. During this procedure, information about different services advertised in the network is gathered (2). This information includes types and descriptions of services and information that the client has to provide in order to be able to use them. Depending on local policies, different resources may have different requirements. Thus, the user agent must select the most suitable service that requires information the user is happy to provide.

The client may need to evaluate the chosen service (3) before using it. The evaluation is performed with the help of local policy (4). Additionally, the client may want to check the quality of the service by collecting opinions of other clients (5). When this is done, and the client wants to continue, she makes a request (6, 7).

When the request is received on the service side, an authorisation agent is created to handle it. This performs risk assessment and checks whether the potential risk is acceptable in terms of the local policy of the resource (9). In doing this, it assesses the potential damage to the user from in terms of the damage it is possible to do to the reputation associated with this identity, including identity loss, or loss of associated values (money may be charged if a credit card number is provided). It then compares this to the potential damage that could be sustained by the resource. It is therefore possible that a user will be declined access when using one of her electronic identities and granted access when using another.

Depending on the circumstances, the authorisation agent may request additional information to be provided by the user (7). However, this may contradict user policy. For example, the user policy may not allow the release of some information from its local network.

When the client provides the information required to use the service, the authorisation agent has to collect evidence that the provided information is correct and that the user who requests the service is indeed the person she claim to be (8).

At this point, the authorisation agent that processes the request has obtained the information on which it will base its authorisation decision. However, it does not yet know whether this information is trustworthy nor whether it has been sent by a legitimate user. The agent does not perform authentication itself. Instead, the authorisation agent delegates this task to third parties that have had previous experience of interactions with this user or different

authorities (10). This runs an automatic credentials discovery (ACD) protocol, which will form the subject of a future paper. However, its basis is that there are different sources available that can be used to verify the user identity, provide information about user's reputation, and verify information provided by user. In pervasive systems, these sources could be distributed over numerous networks and, as a result, might not be capable of working cooperatively. Consequently, the authentication agent must collect their recommendations and combine them to have a reasonable basis for the access control decision it will take. For example, the local profile management server may verify a password's hash sent along with the request; a request signed by user's private key may be verified if one of the parties provides the corresponding public key; there are authorities who can verify credit card numbers; etc. We would like to note that *local* policies and the availability of information dictate the number of steps in this process and proof of identity required to obtain access.

After a request for user credentials is made, the authorisation agent has to collect pieces of knowledge about the user in a secure and private manner. This information must be transforming from heterogeneous opinions into homogeneous data that can be automatically combined and thus allow a decision about user reputation to be made (11). It is worth noting that the result of negotiations between agents is not binary. The negotiations themselves are regulated by a set of fuzzy rules that are dynamically created and reflect local policies. Thus, the authorisation agent may decide that a user's credentials are inadequate to authorise the requested action, e.g. "write", but are adequate to allow another, say "read". The client may accept or decline the offer. Also, she may be willing to give some extra information to get the desired service (for example, some companies require a deposit or a card number if a client does not have a credit history).

After a user's credentials have been collected and evaluated, the authorisation agent decides whether or not to perform the action. If yes, the agent creates an association that is given to client (12). The agent enforces access restrictions by controlling this association. Over the lifespan of an association, authorisation agents perform continuous auditing as a basis for the later (re)assessment of the user's reputation. Audit trails are also used for reactive fraud detection and response (14). Agents perform both misuse and anomaly detection and notify interested parties about any problems. When the action has been completed, the authorisation agent classifies its experience as positive or negative (15) and disseminates updates to user credentials (16). After this, the agent is destroyed, invalidating the client's association. At the client, the user agent evaluates user experience (17) and disseminates service credentials when appropriate (18).

Whilst we have no space to explore this further, it is unreasonable to assume that recommenders always provide accurate testimonies; the system can be subverted both maliciously and as a result of the use of different knowledge management methods or policies. Thus evaluations of testimonies (fraud detection) and agents' ratings are used to maintain overall system integrity by favouring better recommenders.

4. Conclusions

This paper presents a conceptual description of the Distributed Access Control System (ADAM), which is aimed at automation of trust establishment 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 automated trust establishment by gathering information about network entities and later maintenance of trust by constantly controlling information flow and manipulations with network resources. This system facilitates automatic trust establishment and maintenance independently of the type and topology of underlying networks. This provides considerable flexibility and allows ADAM to function in pervasive environments.

5. References

- [1] Reiter, M., Stubblebine, S. "Path Independence for Authentication in Large-scale Systems", 4th ACM Conference on Computer and Communications Security, pp. 57-66, 1997.
- [2] Tarah, A, Huitema, Ch. "Associating metrics to certification paths", ESORICS: European Symposium on Research in Computer Security, pages 175-192, 1992.
- [3] Levien, R. "Attack Resistant Trust Metrics". Draft Ph.D. thesis, 2003.
- [4] Grandison, T. "Trust Specification and Analysis for Internet Applications", Ph.D. Thesis, Imperial College of Science Technology and Medicine, Department of Computing, London, 2001.
- [5] Weeks, S. "Understanding Trust Management Systems", IEEE Symposium on Security and Privacy, pp. 94-105, 2001.
- [6] Blaze, M., Feigenbaum, J., Strauss, M. "Compliance Checking in the PolicyMaker Trust Management System", *Financial Cryptography*, pp. 254-274, 1998.
- [7] Veijalainen, J., Seleznyov, A., Mazhelis, O. "Security and Privacy of the PTP", In Book: *Mobile and Wireless Internet: Protocols, Algorithms, and Systems*, Eds. Makki, K, Pissinou, N, Makki, K., Park, E., Kluwer Publishers, Boston, pp. 165-190, 2003.