Towards an Access Control Mechanism for Wide-area Publish/Subscribe Systems *

Zoltán Miklós

Technical University of Vienna, Distributed Systems Group
Argentinier Straße 8/184-1
A-1040 Vienna, Austria
Z.Miklos@infosys.tuwien.ac.at

Abstract

The publish/subscribe communication model is increasingly considered for implementing middleware infrastructures for widely distributed applications. Scalability issues and routing algorithms of such systems have recently been the focus of intensive research. So far little attention has been given to security and management issues.

In current publish/subscribe systems, malicious publishers can very easily insert bogus notifications which may propagated to a large number of subscribers. Moreover, there is no method to control what notifications the subscribers are authorized to receive.

We describe a method to specify access control policy rules using expressions similar to subscription expressions. These policies define access rules for publish and subscribe functions and screening rules for notifications.

Keywords: publish/subscribe, security, access control

1 Introduction

The publish/subscribe communication pattern is very well suited connecting loosely coupled large-scale applications on the Internet. In this model, receivers of messages express their interest by subscribing to a class of events, and they are asynchronously notified if a sender publishes an event which matches the subscription. In this way the model allows a flexible n-to-m communication among the communicating parties.

Publish/subscribe systems have received increasing attention in the last few years. Both academia and industry researchers are investigating this area [7, 8, 10, 12].

*This work was supported in part by the European Commission under contract IST-1999-10288, project OPELIX (Open Personalized Electronic Information Commerce System).

The common classification scheme of these systems is based on the subscription language. In channel-based systems, a receiver subscribes to notifications sent across a defined channel, whereas in subject-based systems, publishers specify a number of subjects to which clients can subscribe. In content-based systems, the event matching is based on the entire content of the message. In this paper we concentrate on content-based systems.

We also assume a general distributed setting: Publishers send messages to one host in the event-dispatching network. The network routes the message to interested subscribers, who may be registered at another host in the network. The routing mechanism is based on the message content. Figure 1 depicts such a situation.

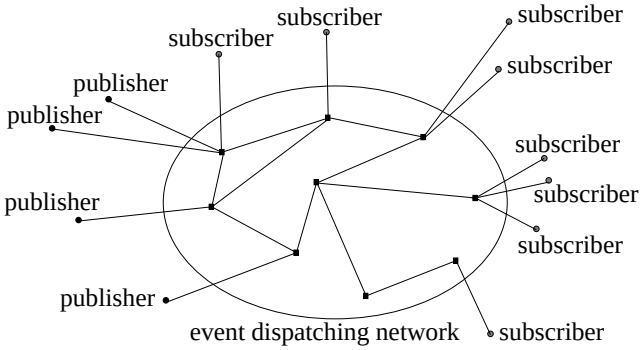

Figure 1. Distributed publish/subscribe system

Current wide-area publish/subscribe systems face serious security problems, which is one of the obstacles to their wider deployment. Wang et al. [13] recently analyzed the security requirements and issues of these systems. Because of the diversity of the scenarios there is a little hope for a

uniform security solution that accommodates all scenarios.

In our access control mechanism the policy rules are based on the content of the message or subscription. We apply subscription and advertisement filters and the covering relations to define the access rights. In our approach the rights to define the policy rules can be delegated to trusted parties who know the semantics of the notifications and subscriptions. In this way effective policy rules can be defined. We grant access rights to credentials. In this paper we focus on how to define the basic control rules. We also present a screening mechanism, which makes it possible to define confidential attributes in the notifications. We use the notation defined in [6].

The remainder of the paper is structured as follows. Section 2 demonstrates some threats that the publish/subscribe systems face without an access control mechanism. Section 3 presents our solution. Section 4 contains related work and Section 5 provides a conclusion and a look at future work.

2 Threats related to lack of access control

In an Internet-scale event based system, the number of publishers and subscribers may be very high. If there is no access control mechanism, all subscribers can subscribe to all event patterns and can receive all published information. Similarly, all publishers can issue events with any content.

These are some attack scenarios which could be prevented with an access control mechanism.

- A malicious publisher can flood the whole network with bogus data. (DoS attack)

- Malicious subscribers can insert fake subscriptions and discard any messages they receive. This attack can be made in an even more coordinated manner if other attackers publish messages on these topics. This will slow down the whole event-distribution network. (Coordinated DoS attack)

- Malicious publishers can issue fake advertisements. This can initiate updates of the content-based routing tables. If subscribers subscribe to these fake content, it causes even more updates. This attack scenario is related to Siena [7], where advertisement messages are defined. (Attack against the routing mechanism)

- In Stock Quotes Dissemination system, where subscribers can specify in which stock quotes they are interested under which conditions, an attacker can send messages under the names of others or with false information. He can, for example, publish false information about the stock quotes of a company and so mislead the subscribers. (Access violation)

- Network event logger is an application which subscribes to all patterns in the network and stores all events. Network loggers intensively consume network resources. A further problem with them in content-based publish/subscribe systems is that logging the events over a long period makes it possible to gain additional information about the network. If a network logger statistically analyzes the event logs and uses a network traffic analyzer, he will be able to identify the anonymous publishers. More generally, he can draw a map of the network and determine which publishers publish information on what topics and which patterns the particular subscribers are interested in. (Identifying anonymous publishers, violating publication confidentiality)

In current publish/subscribe systems there is no access control mechanism defined, so an attacker can very easily insert bogus messages, which then reach large numbers of subscribers. The publish/subscribe service easily can become useless without a suitable control mechanism.

3 Our approach

3.1 Control mechanism

Our goal is to design an access control mechanism which is appropriate for large-scale publish/subscribe systems. Actions for which we would like to authorize principals are publishing an event or subscribing to an event notification. We grant only positive access rights. (It may increase the expressive power of a policy language to define also negative authorizations, but this needs further investigation and is not the focus of this paper.) Without access rights granted by a policy rule, a user is not authorized to publish or subscribe any events. We make the assumption that publishers and subscribers trust their local infrastructure to manage the access rights.

We define a method using subscription or advertisement filters for building groups of notifications and subscriptions to which the policy rules grant access rights. We call these filters access control filters.

We identify authorization subjects by credentials. A credential can be, for example, a digitally signed document or a certificate which contains the user attributes such as name or group membership, or a signed receipt which proves that the user has previously paid for a particular service.

The basic policy rules define access control filters to credentials. Those rules authorize the presenters of these credentials to perform actions which can be related to the access control filters. In this way we can achieve an effective control mechanism, so the access control filters can become the basis of a more complex policy language.

The control mechanism for a publisher works as follows. The publisher sends the message together with his credentials to a host in the event-dispatching network. The host access control component reads the relevant rules from the policy list (which is available locally) and checks whether publishing this action complies with the policy rules. If the publisher is allowed to publish this content, then the message is passed to the message processing component, which starts the event propagation based on the message content. Otherwise the publisher is informed that he is not authorized to send this message (Figure 2). Similarly, the subscriber sends his subscription together with his credentials to the dispatching network host. The access control component on this host checks whether the policy rules and his credentials allow him to insert this subscription. If yes, the subscription is inserted and the subscriber is notified whenever an event matches his subscription.

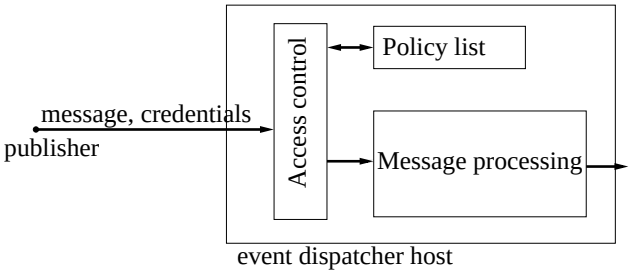

Figure 2. Control mechanism for publishers

3.2 Notation

We use the notation of a concrete realization of the distributed publish/subscribe model [8] to define the policy rules. We use this notation because of its general nature; the results presented here are not limited to that system.

Here we summarize the definitions from [8] and introduce a new relation which we need for our discussion.

An attribute is represented as a triple $\alpha = (type_\alpha, name_\alpha, value_\alpha)$. A constraint is represented as a quadruple $\phi = (type_\phi, name_\phi, operator_\phi, value_\phi)$.

Definition (covers relation) (from [8]): ϕ *covers* α ($\phi \sqsubset_f^n \alpha$) if $type_\alpha = type_\phi \wedge name_\alpha = name_\phi \wedge operator_\phi(value_\alpha, value_\phi)$.

A filter is represented as a conjunction of constraints. A filter covers a notification ($f \sqsubset_S^N n$) if

$$f \sqsubset_S^N n \Leftrightarrow \forall \phi \in f : \exists \alpha \in n : \phi \sqsubset_f^n \alpha. \quad (1)$$

Here we define a new relation:

Definition (strictly covers relation): f *strictly covers* n ($f \prec_S^N n$ for short):

$$f \prec_S^N n \Leftrightarrow f \sqsubset_S^N n \wedge \forall \alpha \in n : \exists \phi \in f : \phi \sqsubset_f^n \alpha. \quad (2)$$

The strictly covers relation is very similar to the covers relation, but it does not allow attributes which have no correspondents in the filter. Examples:

- s_1 = *(string message new_product)*, n_1 = *(string message new_product, integer price 1)*. The relation $s_1 \sqsubset_S^N n_1$ is satisfied, but $s_1 \not\prec_S^N n_1$ since the $price$ attribute has no correspondent in the subscription filter.
- s_2 = *(string message new_product, integer price<5)*, n_2 = *(string message new_product, integer price 1)*. Both the relation $s_2 \sqsubset_S^N n_2$ and $s_2 \prec_S^N n_2$ are satisfied.
- s_3 = *(string message new_product, string color blue)*, n_3 = *(string message new_product, integer price 1)*. $s_3 \not\sqsubset_S^N n_3$ and $s_3 \not\prec_S^N n_3$ because in the notification the value for attribute *color* is missing.

The covering relation can also be defined for subscription filters:

Definition (from [8]): f_1 subscription filter covers f_2

$$f_1 \sqsubset_S^S f_2 \Leftrightarrow \forall n \in N : f_2 \sqsubset_S^N n \Rightarrow f_1 \sqsubset_S^N n. \quad (3)$$

N is the set of all possible notifications. We define here similarly the strictly covers relation for subscriptions.

Definition: f_1 subscription filter strictly covers f_2

$$f_1 \prec_S^S f_2 \Leftrightarrow \forall n \in N : f_2 \prec_S^N n \Rightarrow f_1 \prec_S^N n. \quad (4)$$

For our analysis we need a definition for advertisements which helps to identify the potential notifications.

Definition (from [8]): The set of notifications covered by an advertisement (disjunction of the constraints):

$$a \sqsubset_A^N n \Leftrightarrow \forall \alpha_n \in n : \exists \phi_a \in a : \phi_a \sqsubset_f^n \alpha_n. \quad (5)$$

The cover relation for advertisement filters (from [8]):

$$a_1 \sqsubset_A^A a_2 \Leftrightarrow \forall n \in N : a_2 \sqsubset_A^N n \Rightarrow a_1 \sqsubset_A^N n. \quad (6)$$

We define here the strictly cover relation for advertisements:

$$a_1 \prec_A^A a_2 \Leftrightarrow \forall n \in N : a_2 \prec_A^N n \Rightarrow a_1 \prec_A^N n. \quad (7)$$

Carzaniga has pointed out in [8] that the covers relation defines a partially ordered set of subscription and advertisement filters. Similarly, the strictly covers relation also defines a partial ordering.

3.3 Control rules for publishers

We define here how the policy designer can grant access rights for publishing using publish access control filters.

Granting access rights based on upper bound publish filter: If the policy rules for the credential c of a publisher

define the upper bound publish filter u_c, then he is allowed to publish notifications for which the upper bound filter u_c as a subscription filter covers the notification. For allowed notifications n must satisfy $u_c \sqsubset_S^N n$. If a is an allowed advertisement, then the advertisement must satisfy the relation $u_c \sqsubset_S^S a$.

Examples:

- If only the u_c = *(string message new_product)* upper bound filter is defined for a publisher, then he is allowed to publish the notification *(string message new_product, integer price 10)* but not allowed to publish *(string weather sunny, integer temperature 27)*
- If only the u_c = *(string message new_product)* upper bound filter is defined for a publisher, then he is allowed to issue the advertisement *(string message new_product, integer price $<$ 100)* but not allowed to issue the advertisement *(string weather any, integer temperature any)*

If we also require that the relation $u_c \prec_S^N n$ satisfies, then we call u_c a strict upper control filter.

The designer of the policy may know that the publisher presenting his credential for authorization has knowledge on only a certain number of topics. The designer therefore wish to control him by imposing lower bound access control filters.

Granting access rights based on lower bound publish filters: If the policy rules for the credential c of a publisher define the lower bound publish filter l_c, then he is allowed to publish notifications for which the lower bound publish filter l_c covers as an advertisement filter covers the notification. For allowed notifications, $l_c \sqsubset_A^N n$ must satisfy. If a is an allowed advertisement, then for l_c the advertisement must satisfy the relation $l_c \sqsubset_A^A a$.

Examples:

- If only the l_c = *(string message new_product, integer price $<$ 100)* lower bound filter is defined for a publisher, then he is allowed to publish the notification *(string message new_product, integer price 10, string color blue)* but not allowed to publish *(string weather sunny, integer temperature 27)* or *(string message new_product, integer price 523)*
- If only the l_c = *(string message new_product, integer price $<$ 100)* lower bound filter is defined for a publisher, then he is allowed to issue an advertisement *(string message new_product, integer price $<$ 53)* but not allowed to issue the advertisement *(string message new_product, string color any)*

Strict lower bound filters can prevent more attacks. If a publisher starts to publish messages in a new topic, updates in routing tables, possibly over a large number of nodes are initiated. Thus always publishing in new topics can be seen as an attack against the routing infrastructure. Strict lower bound filters only allow attributes which have a corresponding constraint in the advertisement.

There are effective methods to prove whether the relation $f \sqsubset n$ satisfies [5, 6, 9], so we belive that on the basis of access control filters, an expressive policy language can be defined for which efficient compliance checking exists.

3.4 Control rules for subscribers

Analogously to publishers, we define how access rights to subscribe can be granted based only on subscriptions using the covering relations.

It should be possible for policy designers not to allow very general subscription patterns for all subscribers, in case the application area such security requirements which could not be achieved otherwise (for example, subscription confidentiality or anonymous senders).

Granting access rights based on upper bound subscribe filters: If the policy rules define the upper bound subscribe filter u_c for a subscriber with a credential c, then he is allowed to subscribe to subscriptions for which the upper bound access control filter u_c covers this subscription as a subscription filter $u_c \sqsubset_S^S s$.

Example: If the u_c = *(string message new_product)* upper bound filter is defined for a subscriber, then he is allowed to subscribe to *(string message new_product, integer price 10)* but not allowed to subscribe to *(string weather sunny, integer temperature $>$ 25)*

Further, we can define lower bound filters for subscribers, which enables to prevent a user from being able to subscribe to a very specific condition, if necessary. Matching notifications against large subscription filters with many conditions can be time consuming.

Granting access rights based on lower bound subscribe filters: If the policy rules define the lower bound subscribe filter l_c for a subscriber with a credential c, then he is allowed to subscribe to subscription filters for which the lower bound subscribe filter l_c covers as an advertisement the subscription $l_c \sqsubset_A^A s$.

Example: If only the l_c = *(string message any)* lower bound filter is defined for a subscriber, then he is allowed to subscribe for *(string message new_product)* but not authorized to subscribe for *(string message new_product, integer price $<$ 100)*.

3.5 Information confidentiality for subscribers

Here we present a mechanism which can be used to support information confidentiality for subscribers. This method prevents the disclosure of sensitive information to

non-authorized subscribers, but not to network eavesdroppers or to routing hosts.

The policy designer can control for which subscription filter a subscriber is authorized to subscribe with the method presented in the previous section. On the other hand he has no way to define which attributes of the message must be kept secret to which subscribers, since the notifications received by a subscriber may contain attributes which do not have correspondents in the filter. If we change the notification mechanism slightly, we can make it possible to define the authorized set of attributes. Our idea is that the host in the event-dispatching network which notifies the user not only checks whether the notification is covered by the subscription, but if necessary cuts the attributes according to the access control rights of the subscriber. We call this method screening.

Before we define the screening method formally, we give an example: Let us suppose that a subscriber is allowed to subscribe only for s for which $u_c \prec_S^S s$, where u_c= *(string message any, integer price any)*. So he subscribes to the pattern *(string message new_product, integer price* $<$ *100)*. If a publisher sends the following notification *(string message new_product, integer price 23, string color red)* then the subscriber receives the notification *(string message new_product, integer price 23)*. The attribute *color* will be screened out, since the subscriber is not allowed to read it.

One result of this method is that different subscribers may have different views of the same notification. This is the case if they have different sets of credentials and so different access rights. We consider it acceptable since the subscriber is notified and all information he requested in the message. The subscriber does not even realize the difference.

Definition (screening): Let $f \sqsubset_S^N n$. We say that m is a screening of n related to f if

$$f \prec_S^N m \wedge \forall \beta \in m : \exists \alpha \in n : \alpha = \beta \qquad (8)$$

If the infrastructure provides the screening mechanism, the policy designer can define the allowed set of attributes using the strictly covers relation. With strict upper bound filter u_c he can require a minimum set of attributes: $u_c \prec_S^S s$ for allowed subscriptions. For defining the maximum set of attributes he can use a strict lower bound filter $l_c \prec_A^A s$.

4 Related work

Oasis [1, 2] is a role-based access control architecture designed by the University of Cambridge Computer Laboratory to provide access control for distributed services. Access rights are associated with roles rather than individual principals. Roles are service-specific, the role naming and privilege management is completely decentralized. A principal has to present his credentials at services to activate a role membership. Oasis provides a formal role definition language (RDL) based on Horn clauses in which services can specify the conditions for principals to activate the role. If the principal conforms to the policy, the service issues a role membership certificate which the client presents when he wants to use the service. Because the roles in Oasis can be parameterized, it is possible to express exceptions to the default access control. Role membership certificates are principal-specific, but Oasis can also handle anonymous certificates. The policy rules in Oasis do not rely on the message content, as in our work, since Oasis is not designed for content-based notification service but for general distributed services.

Oasis is built upon the Cambridge Event Architecture. Using event notification, role membership can easily be revoked if some conditions become false. On the other hand publishing and subscribing events may also be built as Oasis-aware services. Cross-domain scenarios are also possible when all domains trust each other. There is no mechanism to involve unknown and therefore untrusted services; however, [2] proposes a certificate issuing and validation (CIV) service.

Wang et al. [13] analyze the security issues and requirements in Internet-scale publish/subscribe systems. The paper also presents a publication control mechanism which is based on a challenging mechanism [14]. The advantage of this scheme is that subscribers can easily establish a filter and subscribe for this challenged publication. With this method a subscriber can receive notifications from legitimate publishers if he previously distributes a secret function with an out-of-band method for them. A problem with this scenario is that only one subscriber initiates the challenging of one (or more) publishers. It is difficult to inform other subscribers, which publication is challenged, if the subscribers do not know each other.

Wang et al. also proposes an application-specific control mechanism to achieve publication confidentiality.

Opyrchal et al. [11] also realize the importance of secure event delivery but they concentrate on the secure distribution of events from the network hosts to the subscribers. They compare different clustering and caching schemes both analytically and empirically by means of simulation. The goal of their work is to reduce the number of encryptions needed so as to increase message throughput.

The Scribe [12] large-scale event notification infrastructure uses also credentials to provide access control. According to the common classification, Scribe is a subject-based system: Scribe nodes may create topics and other nodes can then register their interest in these topics. Credentials are linked to the topics. We analyzed the solution for the more general content-based systems.

5 Conclusion and future work

We identified the need for a scalable publication and subscription control mechanism for wide-area publish/subscribe systems. In this paper we presented a technique to define access control policies. Our approach fits to the publish/subscribe mechanism. This mechanism is only one part of a complete security architecture and can only be effective in cooperation with other security mechanisms.

An area of future research is to analyze how to proceed if the user has more than one credential. A natural next step in the work is to analyze the requirements for the policy language, and further design such a policy language on the basis of access control filters. In our future work we would like to extend the access control component to a trust management engine which supports the delegation of certifying user credentials similar to [3, 4]. The policy rules could also include information about the allowed publication frequency, but this area needs further investigation.

We use the covering relation to build groups of subscriptions and notifications to grant access rights. There are efficient methods of calculating the covering relation so we expect that our mechanism will not require extreme computation resources.

We use user credentials to identify the subjects of authorization, which allows flexible adaptation to the application needs. We also plan to discuss how the principals can obtain the credentials and how the credentials can be realized.

Access control filters can be useful not only at edge routers, but in all network nodes. The access control filters could be propagated together with routing updates. This is useful only if routing updates and the propagation of these filters can be done in a secure way. It must be ensured that no bogus messages can be inserted at routing nodes or between two nodes. At a routing node the access control filters could represent the set of valid subscriptions and notifications. The filters at the routing nodes could be created by merging the filters from neighboring nodes. We will investigate how such a mechanism could improve the security level.

Even if the policy designers know the semantics of the messages, they may require a methodology which helps to identify to which risk level which types of policy rules are applicable.

References

[1] J. Bacon, K. Moody, J. Bates, R. H. C. Ma, A. McNeil, O. Seidel, and M. Spiteri. Generic support for distributed applications. *IEEE Computer*, 33(3):68–76, March 2000.

[2] J. Bacon, K. Moody, and W. Yao. Access control and trust in the use of widely distributed services. In *Proceedings of Middleware 2001.*, pages 300–315, Heidelberg, Germany, Nov. 2001.

[3] M. Blaze, J. Feigenbaum, and A. D. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming*, pages 185–210, 1999.

[4] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 164–173, May 1996.

[5] A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith. Efficient filtering in publish-subscribe systems using binary decision diagrams. In *Proceedings of the 23rd International Conference on Software Engineering. ICSE*, pages 443 – 452, 2001.

[6] A. Carzaniga. *Architectures for an Event Notification Service Scalable to Wide-area Networks*. PhD thesis, Politecnico di Milano, Milano, Italy, Dec. 1998.

[7] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, Aug. 2001.

[8] A. Carzaniga and A. L. Wolf. Content-based networking: A new communication infrastructure. In *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, Scottsdale, AZ, Oct. 2001.

[9] F. Fabret, F. Llirbat, J. Pereira, and D. Shasha. Efficient matching for content-based publish/subscribe systems. Technical report, INRIA, 2000. http://rodin.inria.fr/ pereira/matching.ps.

[10] L. Opyrchal, M. Astley, J. S. Auerbach, G. Banavar, R. E. Strom, and D. C. Sturman. Exploiting IP multicast in content-based publish-subscribe systems. In *Middleware*, pages 185–207, 2000.

[11] L. Opyrchal and A. Prakash. Secure distribution of events in content-based publish subscribe systems. In *Proceedings of the Tenth USENIX Security Symposium*, 2001.

[12] A. I. T. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Third International Workshop on Networked Group Communication, UCL, London, UK, 7-9 November 2001*, pages 30–43, 2001.

[13] C. Wang, A. Carzaniga, D. Evans, and A. L. Wolf. Security issues and requirements for Internet-scale publish-subscribe systems. In *Proceedings of the Thirtyfifth Hawaii International Conference on System Sciences (HICSS-35)*, Big Island, Hawaii, Jan. 2002.

[14] W. Wolf, A. Yasinsac, K. S. Oliver, and R. Peri. Remote authentication without prior shared knowledge. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 159 – 164, 1994.