A Decentralized Treatment of a Highly Distributed Chinese-Wall Policy

Naftaly H. Minsky*
Department of Computer Science
Rutgers University
New Brunswick, NJ, 08903 USA
Phone: (732) 445-2085
Fax: (732) 445-0537
Email: minsky@cs.rutgers.edu

Abstract

Access control (AC) technology has come a long way from its roots as the means for sharing resources between processes running on a single machine, to a mechanism for regulating the interaction among agents (software components, and people) distributed throughout the internet. But despite the distributed nature of the systems being regulated, the conventional enforcement mechanism for AC policies remains basically centralized, where a single (although possibly replicated) reference monitor (RM) is used to mediate the interaction between members of a given community of agents, according to a given policy. This papers demonstrates one of the main drawbacks of centralized AC mechanisms, when applied to distributed systems, and to shows the absence of this drawback under the inherently decentralized law-governed interaction *(LGI) mechanism.*

1 Introduction

Access control (AC) technology has come a long way from its roots as the means for sharing resources between processes running on a single machine, to a mechanism for regulating the interaction among agents (software components, and people) distributed throughout the internet. Distribution introduces several complicating factors to access control, such as insecure communication, heterogeneity, openness, and large scale. Some of the implications of these factors were addressed extensively, and quite successfully, in recent years, by such techniques as: *encryption*, for securing communication; *public-key infrastructures* (PKIs), for scalable key distribution, and for authentication of the identity and roles of principals; *delegation certificates*, for distributed delegation of privileges; and *trust management*, for deciding which rights should be given to the holder of a given set of certificates.

But despite the distributed nature of the systems being regulated, the conventional enforcement mechanism for AC policies remains basically centralized, where a single (although possibly replicated) reference monitor (RM) is used to mediate the interaction between members of a given community of agents, according to a given policy. Although such centralized enforcement is often appropriate, it has some serious limitations, particularly when dealing with *communal policies*, which are not limited to the interaction of a single server with its clients, but govern the interactions among arbitrary members of a distributed community. Specifically, centralized enforcement does not scale well for *dynamic* (or "stateful") communal policies.

The purpose of this paper is to demonstrate this drawback of centralized AC mechanisms, when applied to distributed systems, and to show the absence of this drawback under the inherently decentralized *law-governed interaction* (LGI) mechanism *min00-6,min03-6*. This demonstration will be done via a highly distributed version of the Chinese Wall policy of Brewer and Nash [4], which is introduced in the following section; and by showing, in Section 3, how this policy is formulated and scalably enforced under LGI. Related attempts at distributed versions of the Chinese Wall policy are also discussed in Section 3.

*Work supported in part by NSF grants No. CCR-98-03698

2 The Distributed Chinese Wall Policy

Consider a distributed and heterogeneous collection of information systems, each serving some commercial company. And let these companies be grouped into a disjoint collection of "conflict sets," where each set contains companies that compete with each other in the market place—such as the set of banks, or the set of car dealers. Consider also a distributed collection of financial analysts whose business it is to consult for commercial companies. Now, suppose that the access of these analysts to the companies (i.e., to the information systems serving these companies) is subject to the following policy, to be called CW, for short:

(a) For an analyst to operate under this policy it[1] must authenticate its name, and its status as an analyst, via a certificate signed by a designated certification authority (CA); a company must similarly authenticate its name and the conflict set to which it belongs.

(b) A priori, each analyst can get information from *any* company. But once an analyst gets information from some company `c`, it is not allowed to get information from any other company in the conflict set of `c`.

(c) Copies of messages sent by companies to analysts must be sent to a designated *auditor*.

This is an inherently *communal policy*, which governs a whole community of companies, or company-servers, and their clients. If this policy is to be enforced via the traditional reference monitor, this monitor would have to be replicated for scalability. But replication is very problematic in this dynamic situation, because every state change sensed by one replica needs to be propagated, synchronously, to all other replicas of the reference monitor. specifically, all replicas would have to be informed synchronously about every access of each analyst to every company, lest an analyst sends requests to several companies in the same conflict set, but through different replicas. Such synchronous update of all replica is, of course, possible. But it could be very expensive.

We maintain that this policy calls for a more decentralized approach for the enforcement of distributed AC policies. In the following section we show how this can be done, in a scalable manner, under LGI.

[1] We are using "it" for an analyst, referring to the software component that might be operating on behalf of the human analyst.

3 A Decentralized Treatment of the Chinese Wall Policy

To show how all this works, we introduce here a formalization of our example policy CW, as a law $\mathcal{CW}$ under LGI. For simplicity, this law is left vulnerable to a significant kind of attack. The treatment of this vulnerability under the present LGI will be discussed briefly after the discussion of law $\mathcal{CW}$ itself. It should also be pointed out that policy CW itself is oversimplified, in that it ignores the fact that certificates generally have a limited lifetime. For a treatment of policies that specify what is to be done when a certificate expires or is revoked, and for the formalization of such policies under LGI, the reader is referred to [1].)

Law $\mathcal{CW}$, displayed in Figures 1 and 2, has two parts: *preamble* and *body*. The preamble contains the following clauses: First there is the `cAuthority(publicKey)` clause that identify the public key of the certification authority to be used for the authentication of the controllers that are to mediate $\mathcal{CW}$-messages. This authority is an important element of the trust between agents that exchange such messages—more about which in [2]. Second, there are two `authority` clauses, each of which identifies a certification authority acceptable to this community, one for certifying analysts, the other for certifying companies, identifying the name of each, and the conflict set to which it belongs. Each such CA is identified by its public-key, and is given a local name—"analystCA" and "companyCA" in this case—to be used within this law. Finally, the `initialCS` clause defines the initial control-state of all agents in this community—it is empty in this case.

The body of the law is a list of all its rules, each followed by a comment (in italic), which, together with the following discussion, should be understandable even for a reader not well versed in our language for writing laws.

By Rule $\mathcal{R}1$, one can claim the role of an analyst with a certified name `n`—recoding this information via the terms `role(analyst)` and `name(n)` in its control-state—by presenting an appropriate certificate issued by `analystCA`. Similarly, by Rule $\mathcal{R}2$, a server may authenticate itself as the server of some company `c`, belonging to a conflict set `s`—which would be recorded via the terms `company(c)` and `set(s)` in its control-state—by presenting an appropriate certificate issued by `companyCA`. Note that due to the rest of the law, one cannot function as an analyst or as a company without such terms.

$\mathcal{P}$*reamble:*

```
cAuthority(publicKey).
authority(analystCA, publicKey1).
authority(companyCA, publicKey2).
initialCS([]).
```

$\mathcal{R}1$.
```
certified([issuer(analystCA),
    subject(X),
    attributes([role(analyst),
    name(N)])) :-
  do(+role(analyst)), do(+name(N)).
```

An agent may claim the role of an analyst with name `N`*, by presenting an appropriate certificate issued by* `analystCA`*).*

$\mathcal{R}2$.
```
certified([issuer(companyCA),
    subject(X),
    attributes([C,S])) :-
  do(+company(C)), do(+set(S)).
```

An agent may authenticate itself as the server of company `C`*, belonging to set* `S` *by presenting an appropriate certificate issued by* `companyCA`*.*

$\mathcal{R}3$.
```
sent(X,request(N,C,I),Y)
 :- role(analyst)@CS, name(N)@CS,
  do(forward).
```

A request message by an agent `X` *will be forwarded only if* `X` *has the term* `role(analyst)` *in its control-state, and it must carry its authenticated name.*

$\mathcal{R}4$.
```
arrived(X,request(N,C,I),Y)
 :- company(C)@CS, do(deliver),
  do(+requestedBy(X)).
```

A request for information about company `C` *that arrives at an agent* `Y` *will be delivered to it only if* `Y` *has been certified as serving company* `C`*. Also, a term* `requestedBy(X)` *is added to the control state to record the fact that* `X` *requested information from this company.*

Figure 1. Law $\mathcal{L}_{\mathcal{CW}}$ for Chinese Wall Policy

$\mathcal{R}5$.
```
sent(Y,response(Data,C,S),U)
 :- company(C)@CS, set(S)@CS,
    requestedBy(X)@CS, do(forward).
```

A company-server `Y` *which received a request from an analyst* `X` *can reply to him via a message* `response(D,C,S)`*, where* `D` *is the information requested; and* `C` *and* `S` *are the certified name, and conflict set of the company served by* `Y`*, respectively.*

$\mathcal{R}6$.
```
arrived(Y,response(Data,C,S),X)
 :- not blocked(S)@CS
  do(+blocked(S)),
  do(+permitted(C)), do(deliver),
  do(deliver(Y,[response(Data,C,S),
    X],auditor))
```

A message `response(Data,C,S)` *arriving at an analyst* `X` *would be delivered if there is no* `blocked(S)` *term at the CS of* `X`*, and the term* `blocked(S)` *would be added to the CS of* `X`*, along with the term* `permitted(C)`*.*

$\mathcal{R}7$.
```
arrived(Y,response(Data,C,S),X)
 :- blocked(S)@CS, permitted(C),
  do(deliver).
```

A message `response(Data,C,S)` *arriving at an analyst* `X` *would be delivered if there is* `blocked(S)` *term at the CS of* `X`*, but only if this CS also contains the term* `permitted(C)`*.*

Figure 2. Law $\mathcal{L}_{\mathcal{CW}}$ continuation

By Rule $\mathcal{R}3$, a request message of the form `request(N,C,I)` will be forwarded only if the sender has been authenticated as an analyst (i.e., if it has the term `role(analyst)` in its control-state.) This message must contain the sender's authenticated name `N`, as well as the name `C` of the company for which information `I` is being sought. By Rule $\mathcal{R}4$, when this request arrives at its destination `Y`, it would be delivered *only* if `Y` has been certified as serving company `C`. Also, a term `requestedBy(X)` is added to the control state of `Y`, to record the fact that `X` requested information from it.

By Rule $\mathcal{R}5$ a company-server `Y` which received a request from an analyst `X` can reply to him via a message `response(D,C,S)`, where `D` is the information requested; and `C` and `S` are the certified name and conflict set of the company served by `Y`, respectively.

Finally, the characteristic constraint of the Chinese Wall policy is carried out via the a pair of terms—`blocked(S)` and `permitted(C)`—that can be dynamically attached to a CS of an analyst by a response from a company server, and which determines the disposition of such a response. Specifically, by Rule $\mathcal{R}6$, a message `response(Data,C,S)` arriving at an analyst `X` would be delivered if there is no `blocked(S)` term at the CS of `X`, which, in effect means that `X` did not get any previous responses from a companies that belong to set `S`. Three additional operations are mandated by this rule: (a) the term `blocked(S)` would be added to the CS of `X`, blocking future responses from companies belonging to set `S`; (b) the term `permitted(C)` would also be added to the CS of `X`, permitting (by Rule $\mathcal{R}7$) the delivery of responses from servers of this company, even if the term `blocked(S)` is present; and (c) a copy of the response message is delivered to the distinguished agent `auditor`.

Note that this law does not prevent an analyst

from sending requests to several companies belonging to the same conflict set, nor does it prevent the companies from replying to these messages. But only the first such reply to arrive at the analyst would be delivered, all other replies will be blocked. Note also that such blocking is done strictly locally, and thus scalably.

A Limitation of Law $\mathcal{CW}$, and its Resolution: For simplicity we left an Achilles' heel in this particular law: a possible "double dipping" by an analyst. That is, a single analyst can operate via two (or more) agents under law $\mathcal{CW}$—concurrently or at different times—using the same certificate to authenticate himself as an analyst. Law $\mathcal{CW}$ is not equipped to prevent two incarnations of a single analyst from getting information from different companies in the same conflict-set, which is, of course contrary to the CW policy.

This limitation can be fixed by regulating the membership of the group G of agents operating as analysts under law $\mathcal{CW}$, ensuring the following two properties:

- Group G never includes more than one agent representing an analyst with a given name, as authenticated via a certificate issue by `analystCA`.
- If an analyst has been a member of G and left it, he (or she) will assume his latest control-state when rejoining the group.

The technique for controlling membership under LGI, which can be made to satisfy these properties, has been introduced in [7].

Related Work: There have been several recent attempts at the Chinese Wall policy in distributed context. Kajoth [5] described an implementation of this policy under the Tivoli system, but not in a scalable manner. Tivoli generally uses a replicated reference-monitor for enforcing its access control policies, this works well, and scalably, for regular, static policies, which is what Tivoli usually supports. In addition, Tivoli features what they call an *External Authorization Service*, which can support dynamic policies but is not replicated, and thus is not scalable. It is this centralized enforcer that they use for their implementation of the Chinese wall policy.

Atluri et al. [3] devised a Chinese-Wall-like security model to solve some difficulties with decentralized workflows. But they do not provide a solution to the full gladged distributed Chinese Wall problem itself, and their approach would not scale would they attempt to do that.

Finally, it should be pointed out that the author and his colleague published another solution to the Chinese Wall problem few years ago [6]. That solution was done under a more primitive version of LGI, which required a fairly complex process of initialization of the state of the various agents in a community, and required the law formulating this policy to specify the conflict set explicitly. The present solution is much simpler, and more powerful in some other respects, like the auditing part.

References

[1] X. Ao, N. Minsky, and V. Ungureanu. Formal treatment of certificate revocation under communal access control. In *Proc. of the 2001 IEEE Symposium on Security and Privacy, May 2001, Oakland California*, May 2001.

[2] X. Ao and N. H. Minsky. Flexible regulation of distributed coalitions. In *LNCS 2808: the Proc. of the European Symposium on Research in Computer Security (ESORICS) 2003*, October 2003.

[3] V. Atluri, S. A. Chun, and P. Mazzoleni. A chinese wall secuity model for decentralized workflow systems. In *Proceedings of the Eighth ACM Conference on Computer and Communications Security*, November 2001.

[4] D. Brewer and M. Nash. The Chinese Wall security policy. In *Proceedings of the IEEE Symposium in Security and Privacy*. IEEE Computer Society, 1989.

[5] G. Karjoth. The authorization service of tivoli policy director. In *Proc. of the 17th Annual Computer Security Applications Conference (ACSAC 2001)*, December 2001.

[6] N.H. Minsky and V. Ungureanu. Unified support for heterogeneous security policies in distributed systems. In *7th USENIX Security Symposium*, January 1998.

[7] C. Serban, X. Ao, and N.H. Minsky. Establishing enterprise communities. In *Proc. of the 5th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2001), Seattle, Washington*, September 2001.