An implementation of the Chinese Wall security model using ConSA

Frans Lategan* Martin S Olivier†

March 1998

Abstract

Although security models abound, they are usually an integral part of the system or a totally separate external component. A new architecture, ConSA, allows security models to be developed separately from the system they will protect, and still be integrated seamlessly into the system. Any system using ConSA could then also replace the security model at any time, while retaining the same operating system and applications.

This paper demonstrates two very significant advantages to such an architecture. First, it is flexible enough so that even a non-conventional security model such as the Chinese Wall security model, where access control is based on previously accessed entities, can be implemented using ConSA with ease. Second, in order to present a new security model without needless clutter, the finer details of implementation on a system can be replaced by simply implementing it using ConSA.

1 Introduction

The variety of security models that are appearing on an almost daily basis obviously means that application and system software designers have some tough choices to make when they design a new piece of software. The problem is that a simple security model may not provide adequate protection for their more sophisticated clients, while a complex model may increase costs beyond what is affordable for their smaller clients. In addition, new developments in the security field may render an otherwise good application outdated if its security mechanisms become inadequate. Furthermore, if different systems use incompatible security systems, management of security functionality increases dramatically. ConSA [2] is a recent security model that addresses these issues: ConSA is configurable in the sense that security modules may be 'plugged into' a system that employs ConSA, defining the security characteristics of the system.

Obviously, for a model such as ConSA to be useful, it is necessary to show that it is indeed general enough to support a great variety of current (and future) security models, simply by replacing the 'plug-in' modules. In [2], where ConSA was introduced, it was shown that ConSA can support role-based access control. From the definition of ConSA it is also clear that it will support traditional access control models such as access control lists and capability-based

*Andersen Consulting, PO Box 41168, Craighall, 2024, South Africa, e-mail: frans@9000i.com

†Department of Computer Science, Rand Afrikaans University, PO Box 524, Auckland Park, 2006, South Africa, e-mail: molivier@rkw.rau.ac.za

access control. The current paper extends that work by showing that ConSA can also support the Chinese Wall security model. The fact that ConSA can indeed support this model is important to show ConSA's generality, since the Chinese Wall model differs significantly from other popular security models.

In the Chinese Wall security model access is not granted or revoked by a security administrator, but is instead controlled by the entities already referenced by the particular subject. In this paper we shall be looking at an implementation of the Chinese Wall security model using ConSA. We shall also prove that our implementation of the Chinese Wall security model on ConSA is correct.

The remainder of this paper is structured as follows: Section 2 contains background information. Section 3 describes the ConSA model in more detail; Section 4 describes the Chinese Wall security model in more detail, then, Section 5 presents an implementation of the Chinese Wall using ConSA, and lastly, Section 6 concludes this paper.

2 Background

One of the biggest problems with computer security, can be stated as follows: how much security is enough? There is no simple answer to this question today, and probably never will be. The reason that this poses a problem, is that the more security is needed, the greater the cost. Further, having a secure operating system does not imply that all applications running on that operating system are secure, and the converse does not apply either.

It is precisely in this area that ConSA provides a simple solution. ConSA allows the operating system and application developers to concentrate on their task, without worrying about the level or type of security required in the final product. The level and type of security required by the end user is then chosen by purchasing the appropriate security plug-ins, and then the operating system as well as the applications are protected.

In short, ConSA consists of plug-in modules with a standard set of methods which have to be defined according to the security model used. In order to implement any security model using ConSA, two steps are of importance. Firstly, the standard methods have to be defined, and secondly it has to be shown that the required post conditions apply after execution of these methods.

In order to refer to the objects in this paper in an unambiguous way, the following conventions will be adopted: An entity is an object in the system that requires protection by the security model, such as a database record or a file. A subject is an object in the system that requires access to some entity, such as a user or an application program. In order to access an object, a message will be sent from the subject to the object, via the security model.

Each entity and subject in the system has a label associated with it that describes their security information. In this paper the labels consist of a set of special tags.

The Chinese Wall security model (see [5]) which is used in this paper is an interesting model in the sense that access to entities is not granted or revoked by a security administrator, but is instead limited to entities already accessed in discrete conflicts of interest classes - if entities e_1 and e_2 belong to the same conflict of interest class C, then access to e_1 is only granted to a subject if that subject has not yet accessed e_2, and vice versa. Flow control is also required. Flow control is a mechanism which prevents a subject with read access to a particular entity e_1 to write information from that entity e_1 to another entity e_2 where a subject without read access to e_1 could read it. This is the $\star$-property of Bell and LaPadula (see [1]). It is required by the

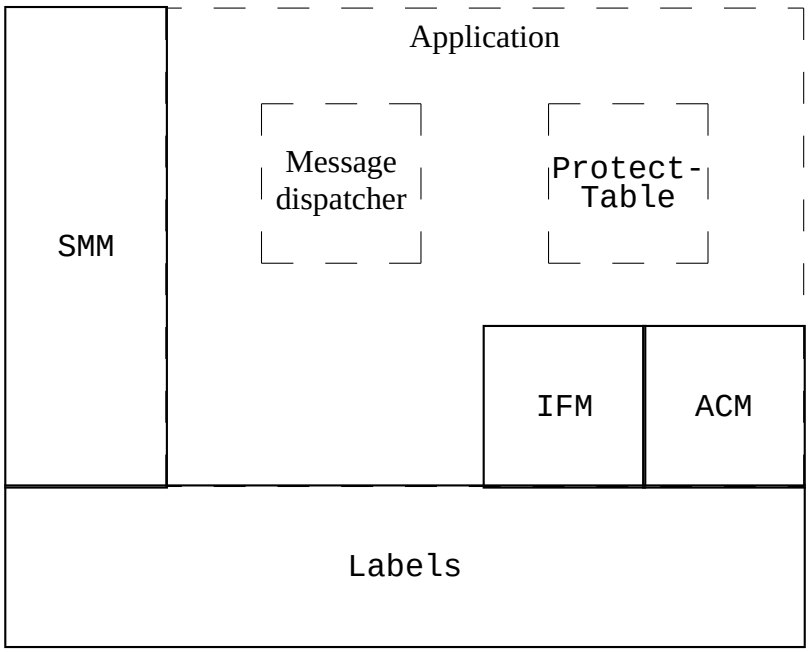

Figure 1: ConSA

Chinese Wall model since a subject s_1 has to be prevented from writing information from an entity e_1 not normally accessible to another subject s_2 to an entity e_2 to which subject s_2 has access.

In this paper, we shall confine ourselves to discuss the implementation of the Chinese Wall security model using ConSA, and therefore not discuss ways to enforce the access control, or actually implement ConSA on an existing operating system. See [4] for more information on other security models.

3 A description of ConSA

ConSA [2] defines an object-oriented security model which is independent of the operating system and applications. It is a configurable, plug-in model, which can be adapted to the required level of security. Instead of having an operating system supply part of the security, and having applications such as databases with their own security measures, ConSA defines a way in which both the operating system and the applications use a common set of security plug-ins.

This system has obvious advantages in that DBMS developers can focus on the DBMS, operating system developers can focus on the operating system, and security developers can focus on the security. A particular operating system or application could also then conceivably be protected by a security model that was not even in existence when they were purchased.

ConSA is also of importance for another very good reason. Using ConSA, one can develop any security model without worrying about the details of the actual implementation thereof. Much like the first NP-complete problem made it much easier to prove any other problem NP-complete by proving them equivalent (see [3]), ConSA offers a quick way to present a new security model by simply giving an implementation for it using ConSA.

Figure 1 shows an outline of ConSA, as it will typically be used in an object-oriented oper-

ating system. The *Labels* module defines the label classes; the *Information Flow Module* (IFM) handles the flow control, and *Authorisation Control Module* (ACM) controls subject access to entities. The *Subject Management Module* (SMM), used by the security manager, is not really required for the Chinese Wall model used here, since the primary subject access restrictions and modifications are automatically handled by the ACM module (this becomes clear later, when the `CheckAccess` method is defined).

ConSA consists of a number of methods which have to be defined for a particular security model, with certain conditions which must apply after their execution to present a consistent security screen.

The more important components of ConSA will now be covered briefly. A more detailed description can be found in [2].

3.1 Fundamental concepts and notation

The following terms will also be used henceforth:

- $\mathcal{E}$ will denote the set of all entity labels; L_e will denote the label associated with entity e.
- $\mathcal{S}$ will denote the set of all subject labels; L_s will denote the label associated with subject s.
- If a subject label L_s will grant subject s access to entity e protected by L_e, it will be denoted by writing $L_s \preceq L_e$.
- $auth_0(L_s) \subseteq \mathcal{S}$ will denote the set of subjects directly associated with the subject label L_s. It is also possible for a subject to be indirectly associated with a subject label through inheritance. In the classical military example, top secret clearance automatically inherits all the access rights of the secret clearance level.
- $auth(L_s) \subseteq \mathcal{S}$ will denote the set of subjects directly or indirectly associated with the subject label L_s.
- $subj(L_e) \subseteq \mathcal{S}$ will denote the set of subjects authorised to access an entity protected with $L_e \in \mathcal{E}$.
- $ent(L_s) \subseteq \mathcal{E}$ is defined as the set of all entity labels for which access will be granted when presented with the subject label L_s.

¿From the above it is clear that $subj(L_e) = \{s \in \mathcal{S} \mid s \in auth_0(L_s), L_s \preceq L_e\}$. It also follows that $ent(L_s) = \{L_e \mid L_s \preceq L_e\}$.

The core of ConSA is made up of the security methods presented below:

- `NoAccess(dest:` L_e`)` modifies the entity label L_e such that no subjects have access to the entity e. After the execution of `NoAccess`, $subj(L'_e) = \emptyset$.
- `GrantAccess(dest:` L_e`, par:` L_s`)` modifies L_e such that subject s can access entity e. After the execution of `GrantAccess`, $subj(L'_e) = subj(L_e) \cup auth(L_s)$.
- `RevokeAccess(dest:` L_e`, par:` L_s`)` modifies L_e such that subject s can no longer access entity e. The formal definition of `RevokeAccess` is omitted, since it will not be used in the Chinese Wall model.

- `CheckAccess(dest:` L_e`, par:` L_s`)` sends back $true$ or $false$, based on whether subject s can access entity e or not. Formally, `CheckAccess` returns $true$ if $L_s \preceq L_e$.
- `Dominates(dest:` L_{e_1}`, par:` L_{e_2}`)` sends back $true$ or $false$, based on whether L_{e_1} dominates L_{e_2}, that is, all subjects with access to L_{e_1} have access to L_{e_2}. `Dominates` (also written as $L_{e_1} > L_{e_2}$) returns $true$ if $subj(L_{e_1}) \subseteq subj(L_{e_2})$.
- `IncreaseSens(dest:` L_{e_1}`, par:` L_{e_2}`)` modifies L_{e_1} so that only subjects that could previously access both L_{e_1} and L_{e_2}, can access L_{e_1}. Formally, after the execution of `IncreaseSens`, $subj(L'_{e_1}) = subj(L_{e_1}) \cap subj(L_{e_2})$.
- `InitSubject(dest:` L_s`)` initialises subject labels.

To provide an insight into the flexibility of this model, we shall now demonstrate the implementation of the Chinese Wall security model on a system equipped with ConSA.

4 A description of the Chinese Wall Access Control Model

The Chinese Wall access control model is a model where all subjects have access to all entities in the system when they are created. Restrictions are then added as subjects access entities, until at some point a subject will only have access to those entities already accessed previously.

Entities are grouped into several groups. As soon as a subject has accessed an entity, access is revoked for that subject to all other entities in the same group as the entity that was accessed.

As an example, this model could be used to govern access to documents by a consultant in a consulting firm. We now consider this in more detail.

4.1 Access Control

In principle, a consultant has access to all documents in the firm. As soon as the consultant has read a document on say, insurance firm A, access to all documents relating to insurance firms other than A is revoked. This is done to prevent possible conflicts of interest and decisions based on confidential information about competitors of firm A who may also be clients of the consulting firm. The consultant should still have access to other documents about firm A, and of course, all documents that do not contain information about insurance firms.

4.2 Flow Control

When writing information, the newly created document should in some way be identified as being from an insurance firm, and that it contains information about firm A. All consultants with access rights to the information about firm should be granted rights to the new document. However, all consultants having accessed information about any insurance firm other than A should not have access to the new document.

5 The Chinese Wall using ConSA

This section demonstrates the use of ConSA to implement the Chinese Wall access control model.

We shall make the following assumptions:

1. All entities containing information about the same company will have the same label.
2. When a subject has accessed an entity in a group, access to other entities in the group with different labels will be revoked indefinitely.

5.1 Notation

Recalling our consulting firm example, we have to be able to distinguish between entities containing information from different companies in a conflict of interest group. We must also be able to distinguish between different conflict of interest groups.

To enable this distinction, we shall use tuples, or tags, denoted by $t_{i,j}$ where i identifies the conflict of interest class that this tag belongs to, and j denotes the specific member of that conflict of interest class. Both indices will normally start at 1, with 0 reserved for special categories defined below.

A group will contain a set of tags, with the first index of each tag the same as the group number. Each group will represent a distinct conflict of interest class. Each subject label will also consist of a set of tags, and each entity label will also consist of a set of tags denoting group membership, or the special tag $t_{0,1}$.

A subject with a subject label containing a tag $t_{i,j}$ must therefore be refused access to all entities having a tag $t'i, k$, $k \neq j$ in their entity label.

The following notation will be used:

- g_i will be used to denote a group i. $\mathcal{G}$ will denote the set of all groups.
- Each group g_i will contain a special tag $t_{i,0} \mid (\forall L_e \in \mathcal{E})(t_{i,0} \notin L_e)$. This tag could then be used to revoke access to all entities in that particular group, by inserting it into all subject labels where access has to be revoked — if $t_{i,0} \in L_s$, it is clear that $\forall k, k \neq 0$, access to an entity with $t_{i,k}$ in its entity label will be refused to the subject with subject label L_s. Such a subject would also not be granted access to an entity with tag $t_{i,0}$, since by definition, there can be no such entity.
- There will also be a special group $g_0 = \{t_{0,0}, t_{0,1}\}$ which can be used to create entities to which no subject has access by inserting $t_{0,0}$ into all subject labels, and assigning $t_{0,1}$ to the label of such an entity. No subject label may contain $t_{0,1}$, and by the definition of `InitSubject`, all subjects will contain $t_{0,0}$. Also, there will be no entity label that contains tag $t_{0,0}$, by the definition of tags of the form $t_{i,0}$. Since there is no method which removes tags from a subject label, it is clear that no entity with an entity label containing tag $t_{0,1}$ can be accessed by any subject.
- $\mathcal{G}(L_e)$ will denote the group or groups to which the entity with label L_e belongs.
- Each entity label can contain at most one tag from every group.

5.2 Implementation

In this section we now implement the Chinese Wall security model using ConSA. We start with a definition for dominance, that is, when will access to a particular entity be granted.

If $L_s = \{l_1, l_2, \ldots, l_n\}$ then $L_s \preceq L_e \Leftrightarrow (L_s \cap \mathcal{G}(L_e) \backslash L_e = \emptyset)$

It then follows that

$subj(L_e) = \{s \in \mathcal{S} \mid L_s \cap \mathcal{G}(L_e) \backslash L_e = \emptyset\}$

Also,

$ent(L_s) = \{L_e \in \mathcal{E} \mid (\mathcal{G}(L_e) \backslash L_e \cap L_s = \emptyset)\}$

It is trivial to see that $auth(L_s) = auth_0(L_s)$.

In certain cases it might be necessary to have an entity to which no subject in the system has access. To set the access rights for such an entity, `NoAccess` is used, and is defined as follows:

`NoAccess` will replace L_e with L'_e where $L'_e = \{t_{0,1}\}$. This, together with the definition of `InitSubject` below, will ensure that $subj(L'_e) = \emptyset$.

All subjects in the system will be initialised on creation with the method `InitSubject`. `InitSubject` will replace L_s with L'_s where $L'_s = \{t_{0,0}\}$.

We are now in a position to prove the correctness of the definition of `NoAccess`:

Theorem 5.1 `NoAccess` *modifies an entity label* L_e *such that* $subj(L'_e) = \emptyset$ *where* L'_e *indicates the entity label after the execution of* `NoAccess`

Proof: After the execution of `NoAccess` on an entity label L_e, $L'_e = \{t_{0,1}\}$. By the definition of `InitSubject`, $\forall L_s \in \mathcal{S}, t_{0,0} \in L_s$.

Further, by definition, $g_0 = \{t_{0,0}, t_{0,1}\}$. It then follows that $\forall s \in \mathcal{S}, L_s \cap \mathcal{G}(L'_e) \neq \emptyset$. This implies that $subj(L'_e) = \emptyset$. □

The core of the Chinese Wall security model is the `CheckAccess` method, since it not only verifies a subject's rights to access an entity, but it also modifies a subject's label so that that particular subject is prevented from subsequently accessing a different entity from the same conflict of interest class. In order to do this, `CheckAccess` will add the tags in an entity label to a subject label if access to the entity is granted for a subject.

`CheckAccess` will return $true$ if $(\mathcal{G}(L_e) \backslash L_e) \cap L_s = \emptyset$. If `CheckAccess` returned true then L_s will be replaced with L'_s, where $L'_s = L_s \cup L_e$. This is the most important method of the Chinese Wall security model, as `CheckAccess` will also modify the subject labels as access to subjects are granted.

We now prove that after accessing a particular entity, a subject can no longer access any other entity from the same conflict of interest class.

Theorem 5.2 *If* `CheckAccess` *returns* $true$ *for an entity label* L'_e *and a subject label* L_s*, then after execution* $ent(L's) \cap (\mathcal{G}(L'_e) \backslash L'_e) = \emptyset$.

Proof: After the execution of `CheckAccess`, if `CheckAccess` returned true, $L'_e \in L'_s$.

This implies that $\forall L''_e \in (\mathcal{G}(L'_e) \backslash L'_e), \mathcal{G}(L''_e) \backslash L''_e \cap L'_s = L'_e \neq \emptyset$.

By definition, $ent(L_s) = \{L_e \in \mathcal{E} \mid (\mathcal{G}(L_e) \backslash L_e \cap L_s = \emptyset)\}$.

Thus, $ent(L's) \cap (\mathcal{G}(L'_e) \backslash L'_e) = \emptyset$ □

Flow control is of course required in the Chinese Wall model, since it would defeat the purpose if one subject could copy an entity (or reproduce part thereof) in another conflict of interest class, so that a particular subject could access information it should not have had access rights to. The two methods required for flow control are defined below.

For a complete description of their use, see [2], but in short, a subject is only allowed to write information read from an entity with label L_{e_1} to an entity with label L_{e_2} if L_{e_2} dominates L_{e_1}. `IncreaseSens` can be used to ensure that L_{e_2} will dominate L_{e_1}.

`Dominates` sent to an entity label L_e with entity label M_e as parameter, will return $true$ if $M_e \subseteq L_e$.

A subject e dominates another subject f if it has access to at least all of the entities that f has access to.

Theorem 5.3 *If* `Dominates` *returns* $true$ *for a two entity labels* L_e *and* M_e*, with* L_e *as the first parameter and* M_e *as the second, then* $subj(L_e) \subseteq subj(M_e)$.

We now prove the definition consistent with the ConSA model.

Proof: Per definition, if `Dominates` returns $true$, then $M_e \subseteq L_e$.

Since $M_e \subseteq L_e$, it follows that $\mathcal{G}(M_e) \subseteq \mathcal{G}(L_e)$. Further, since each entity label can contain at most one tag from each group, $\mathcal{G}(M_e) \backslash M_e \subseteq \mathcal{G}(L_e) \backslash L_e$

Also, per definition, $subj(L_e) = \{s \in \mathcal{S} \mid L_s \cap \mathcal{G}(L_e) \backslash L_e = \emptyset\}$.

Since $\mathcal{G}(M_e) \backslash M_e \subseteq \mathcal{G}(L_e) \backslash L_e$, it implies that $\mathcal{G}(L_e) \backslash L_e$ will exclude at least as many elements of S as $\mathcal{G}(M_e) \backslash M_e$, from which it follows that $subj(L_e) \subseteq subj(M_e)$ □

`IncreaseSens` sent to one entity label with another as a parameter is used to ensure that the entity label in question will dominate the entity label sent to the method as a parameter.

`IncreaseSens` sent to an entity label L_e with entity label M_e as parameter, will change L_e into L'_e such that only subjects that have access to both L_e and M_e will have access to L'_e. This is done as follows:

$L'_e = L_e \cup M_e$. Because of the constraints of the Chinese Wall model, `IncreaseSens` can not be applied to an entity label L_e with parameter M_e if after execution

$\exists g_i \in \mathcal{G}(L'_e) \mid (\mid g_i \mid) > 1$. This means that if there is a group $g_i \in \mathcal{G}(L_e)$ and the same group $g_i \in \mathcal{G}(M_e)$, then $g_i \backslash L_e = g_i \backslash M_e$ if `IncreaseSens` is defined.

The proof follows.

Theorem 5.4 *If* `IncreaseSens` *is sent to a target label* L_e *with a parameter label* M_e*, then* $subj(L'e) = subj(L_e) \cap subj(M_e)$.

Proof:

Per definition of `IncreaseSens`, $L'_e = L_e \cup M_e$. It then follows that $\mathcal{G}(L'_e) = \mathcal{G}(L_e) \cup \mathcal{G}(M_e)$.

Since each entity label can contain at most one tag from each group, and `IncreaseSens` is only defined if for every group g_i in both $\mathcal{G}(L_e)$ and $\mathcal{G}(M_e)$, $g_i \backslash L_e = g_i \backslash M_e$,

$\mathcal{G}(L'_e) \backslash L'_e = (\mathcal{G}(L_e) \backslash L_e) \cup (\mathcal{G}(M_e) \backslash M_e)$.

Also, per definition, $subj(L_e) = \{s \in \mathcal{S} \mid L_s \cap \mathcal{G}(L_e) \backslash L_e = \emptyset\}$,

from which it follows that $subj(L'e) = subj(L_e) \cap subj(M_e)$. □

5.3 Further notes

Some of the standard methods defined in ConSA are not used in the basic Chinese Wall security model. They are presented here for the sake of completeness.

`GrantAccess` sent to an entity label L_e will remove $t_{0,1}$ from the entity label if it is there, as well as remove $t_{\mathcal{G}(L_e),0}$ from the subject label L_s. Note that in cases where $L_s \cap \mathcal{G}(L_e) \backslash t_{\mathcal{G}(L_e),0} \neq \emptyset$, access will still not be granted to the entity. Therefore this version of `GrantAccess` will be renamed `GrantAccess0`. Another version, `GrantAccess*` will unconditionally grant access to the entity by removing all $L_{e'} \in \mathcal{G}(L_e)$ from the subject label

L_s. In general, though, `GrantAccess` will not be used with the Chinese Wall security model. It is presented here for informational purposes only.

`RevokeAccess` will add $t_{\mathcal{G}(L_e),0}$ to L_s and remove L_e from L_s. This means that no entity from the group $\mathcal{G}(L_e)$ could be accessed with L_s anymore. Once again, `RevokeAccess` will not be used with the Chinese Wall security model in its basic form.

6 Conclusion and areas for further study

In this paper we have presented a brief outline of the ConSA configurable security architecture, and noted its significance in extending the life of software products by allowing the security model to be easily changed. In order to demonstrate the diversity and suitability of ConSA, an unconventional security model, the Chinese Wall, was implemented using ConSA, showing that it is indeed possible to implement the Chinese Wall using it.

Possible areas for future study include generalising the implementation of the Chinese Wall model using ConSA to a combination of Chinese Wall and role-based security, such as might be found in a real consulting firm (Partners having more access rights than Managers, which in turn have more access rights than Consultants). It could even be of interest to implement even more esoteric security models on ConSA, further demonstrating its flexibility.

References

[1] D. E. Bell and L. J. LaPadula, "Secure computer system: unified exposition and Multics interpretation", *Rep. ESD-TR-75-306*, March 1976, MITRE Corporation.

[2] M. S. Olivier, "Towards a Configurable Security Architecture", *Data and Knowledge Engineering*, 38, 2, 121–145, 2001.

[3] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman, San Fransico, CA, 1979.

[4] C.P. Pfleeger, "Security in Computing", Prentice-Hall, 1989.

[5] R. S. Sandhu, "Lattice-Based Access Control Models", *Computer*, November 1993, 9–19.

FA Lategan and MS Olivier, "An implementation of the Chinese Wall security model using ConSA," Internal Report, Department of Computer Science, Rand Afrikaans University, 1998

Source: http://mo.co.za