A Security Policy Model for agent based service-oriented architectures

Eckehard Hermann
eckehard.hermann@fh-hagenberg.at

Department of Secure Information Systems
Upper Austria University of Applied Sciences, Austria

Abstract. During the last years service oriented architectures (SOA) have gained in importance, when looking at today's implementation of business processes. A SOA is a loosely coupled system of services, where a service is implemented by an agent. The protection of information and data objects and their well-directed flow are essential for the success of enterprises, which also applies to the communication inside a SOA. To guarantee an approved protection of data objects and to prevent an illegal information flow, approved security policy models are chosen that are suitable for the considered use case. The Limes Security Model [1] is based on a not necessarily symmetric, not necessarily reflexive and not necessarily transitive conflict of interest relation. The model is introduced for pure subject/object relationships, where agents are not taken into account. The current paper extends the Limes Security Model by the support of agents, suitable for the use in a SOA.

Keywords: security models, service-oriented architectures, principal agent theory, information flow control

1 Introduction

Service oriented architectures (SOA) have gained in importance by the implementation of business processes as a dynamic and loosely coupled system of services. In [8] Burbeck defines the term *service-oriented* for

> architectures that focus on how services are described and organized to support their dynamic, automated discovery and use [...] to work easily, flexibly, and well together, services must be based on shared organizing principles that constitute a service-oriented architecture (SOA), the architectural concept behind Web services.

Such kinds of services are used in business-to-business scenarios, where the services communicate with each other by sending and receiving messages via a request/response protocol. A client invokes a service by sending a request and providing the necessary data that the service needs to process in order to preparing and sending the response. A service is implemented by an agent, a program that acts on behalf of a person or an organization [2] at runtime. The described

situation is a typical outsourcing scenario, where a client outsources one or more tasks to a service instead of implementing them himself. As described by Pratt and Zeckhausen in [9]

> Whenever one individual depends on the action of another, an agency relationship arises. The individual taking the action is called the agent.

The problem described by Pratt and Zeckhausen in [9] is based on an asymmetric relationship, because of an informational advantage of the agent compared to the client. In many cases the client (application) provides data of some sort of critical nature to the agent. The problem lies in the uncertainty of the client, not knowing whether the agent might misuse this data. To guarantee an approved protection of the client data and to prevent an illegal information flow, an approved security policy model has to be chosen that is suitable for the considered use case. The security policy model defines the rules and policy that have to be implemented by any kind of read and write access of all the participants to prevent provable an illegal information flow.

The current paper gives an introduction to the Limes Security Model, discusses the problems, where the model is applied to agents and presents an extension of the Limes Security Model by the support of agents, suitable for the use in service oriented architectures.

2 State of the Art

2.1 Agent based Service-Oriented Architectures

A service-oriented architecture is a loosely coupled system of services. Resources are made available as independent services, with a high degree of abstraction of the underlying platform [10], which can be dynamically integrated into new applications. Resources are made available as independent services, which can be dynamically integrated into new applications.

A service is characterized by its abstract set of functionality that it provides and which is implemented by an agent, a program that acts on behalf of a person or an organization [2]. For the rest of this paper agents are defined according to the characterization of [2]. An agent

- is a computational resource,
- has an owner that is a person or organization,
- may realize zero or more services,
- may request zero or more services. [2]

2.2 Security Models

Different security policy models like the one from Bell and LaPadula, are a formalization of a military security model in [5] or they address the integrity of data objects in commercial transactions, as stated by Clark and Wilson [6].

In 1989 Brewer and Nash presented their Chinese Wall Security Policy model (CWSP model) that is based on conflict of interest classes [3] and nearly one year later Lin showed some limitations of the CWSP model and presented in [4] a modified version, called the Aggressive Chinese Wall Security Policy model (ACWSP model).

The Chinese Wall and the Aggressive Chinese Wall Security Policy model As part of the Chinese Wall Security Policy model Brewer and Nash defined in [3] a hierarchically arranged filing system, where, like Loock and Eloff summarized in [7], on the lowest level, individual items of information are considered, each concerning a single corporation. At the intermediate level, all objects, which concern the same corporation are grouped together and form the company dataset. And on the highest level all company datasets are grouped, whose corporations are in competition. The highest level is called the conflict of interest classes [3]. If a subject intends to access an object, access is only granted if the object requested

- is in the same company dataset as an object already accessed by that subject or
- belongs to an entirely different conflict of interest class [3].

Lin showed in [4] that Brewer and Nash implicitly assume that the conflict of interest is an equivalence relation. He showed that the conflict of interest is a binary relation but not in general an equivalence relation. Lin assumed that the conflict of interest relation is symmetric but non-reflexive and non-transitive, except under special conditions.[4]

Let us recollect some of our earlier considerations [1]: Because each organization defines its own security policy, symmetric conflict of interest classes are not the default in the real world of business. When an organization A defines a conflict with organization B, it should be independent of the definition of conflicts that B creates on its own behalf.

Example 1. Let O = {Company A, Opensource Community} and let CIR = "in conflict with". Let the Company A be "in conflict with" the Opensource Community because the business model of the Company A depends on licensing their software, which would imply the protection of their own knowhow against competitors. If CIR were symmetric and Company A would be "in conflict with" the Opensource Community, it would imply that the Opensource Community is "in conflict with" Company A, which is obviously not the case. [1]

3 Related Work

In the recent past security policy models for web services environments or Workflow Management Systems have been developed by Hung et al or by Hsiao and Hwang. Hung et al extended the CWSP model in [12] into specifying and implementing conflict of interest assertions in WS-Policy for Web Services enabled

environments. Hsiao and Hwang implemented the CWSP model into a Workflow Management Systems in [11].

Debasish and Bijan described in [13] an AAA (Authentication, Authorization and Accounting) security model of service oriented computational grids. The model is not a provable formalized model. It has been modeled and implemented by using a token based authentication, based on Kerberos or a PKI and implementing XML Signature and XML Encryption for integrity and privacy.

Wu et al adopt in [14] the Chinese Wall policies to address the problems of insecure information flow inside a cloud and to resolve the conflict-of-interest issues for service providers in clouds.

The Limes security model, introduced in [1], extends the CWSP model and is based on a not necessarily symmetric, not necessarily reflexive and not necessarily transitive conflict of interest relation.

4 The Limes Security Model

The Limes Security Model as defined in [1] works on the assumption that an object does only stay in conflict with an object and does not stay in conflict with a subject. This concludes that the information flow between objects has to be controlled, which can be done by controlling the write accesses of the subjects. A conflict of interest is not necessarily symmetric, not necessarily reflexive and not necessarily transitive. Each object is able to express its individual conflict of interest by the definition of its own time depending Conflict Function and Conflict Of Interest List. In [1] the Conflict Function and Conflict Of Interest List are defined as follows:

Definition 1. *Let S be a set of subjects and let O be the set of all known objects. $O_t \subseteq O$ is the set of all available objects at instant of time $t \in \mathbb{N}$.*

Definition 2 (Conflict Function). *For each instant of time $t \in \mathbb{N}$, all objects $i, j \in O_t$ and each instant of time $l \in \mathbb{N}$ with $l \leq t$ and where l is the instant of time, where a read access to object i has been performed, let ${}_iCIL_l^t$ be the Conflict Of Interest List and let ${}_iNCL_l^t$ be the Non Conflict Of Interest List of the object i depending on the read access to the instant of time l with the following properties [1]:*

- ${}_iCIL_l^t \cup {}_iNCL_l^t = O_t$.
- ${}_iCIL_l^t \cap {}_iNCL_l^t = \{\}$.
- $i \in {}_iNCL_l^t$.
- *If $j \in {}_iNCL_l^t$, then at instant of time t the object i is not in conflict with object j in relation to the read access at instant of time l.*
- *If $j \in {}_iCIL_l^t$, then at instant of time t the object i is in conflict with object j because of the read access at instant of time l.*

A function $f_i^t : O_t \times \mathbb{N} \to \mathbb{Z} \cup \{\infty\}$ is called Conflict Function of object i if f_i^t has the following property:

$$f_i^t(j,l) = \begin{cases} < 0, & \text{if } {}_iCIL_l^t = \{\}, \\ 0, & \text{if } j \in {}_iNCL_l^t \wedge {}_iCIL_l^t \neq \{\}, \\ > 0, & \text{if } j \in {}_iCIL_l^t. \end{cases} \tag{1}$$

All data that is owned by an individual object or subject, is grouped together and defines the dataset of the individual object or subject. If a subject performs a read access to an object, we assume that it reads the whole dataset of the object. If the subject performs a write access to an object, we assume that it writes its own dataset completely into the dataset of the object.

Each subject owns its own Read Access History, containing the information about the sources of the data in its own dataset and the instance of time, when the data has been read from the individual objects. The Read Access History ${}_sH^t$ is the set of tuples (j, t') of objects $j \in O_t$ that have been read accessed by the subject $s \in S$ in the past and the instant of time $t' \in \mathbb{N}$ of the individual read access.

In addition to the Read Access History of the subjects, each object also owns a history, called the Dataset Actuality of the object. The Dataset Actuality ${}_iA^t$ of an object $i \in O_t$ is the set of tuples (j, t'). The objects $j \in O_t$ have been read accessed by a subject $s \in S$ in the past and the read dataset has been written to the object i. t' is the instance of time the read access has been performed at j. The Dataset Actuality contains the information about the sources and actuality of the data contained by the dataset of i. In addition to the Dataset Actuality each object owns an Actuality Function as defined by Definition 3.

Definition 3 (Actuality Function). *For each instant of time $t \in \mathbb{N}$ and all objects $i, j \in O_t$ let $t' \in \mathbb{Z}$ be the instant of time, where the read access at object j has been made and where the read dataset has been written to i later. The function $a_i^t : O_t \to \mathbb{Z}$ is called the Actuality Function of i, if a_i^t has the following property [1]:*

$$a_i^t(j) = \begin{cases} t', & \text{if } (j,t') \in {}_iA^t, \\ -1, & \text{otherwise.} \end{cases} \tag{2}$$

Example 2. Figure 1 (*a.*) shows the instant of time t of an example, where the subjects s_1 and s_2, pictured as circles, have not accessed any of the objects o_1 and o_2, which are pictured as rectangles. Therefore the Read Access Histories of the subjects and the Dataset Actualities of the objects are empty.

If a subject performs a read access to an object, it adds the dataset of the object to its own dataset. Additionally it merges its own Read Access History with the Dataset Actuality of the object in a way that its own Read Access History contains the entries of both histories afterwards. The entry with the more recent instance of access time is selected in case of a collision. Additionally

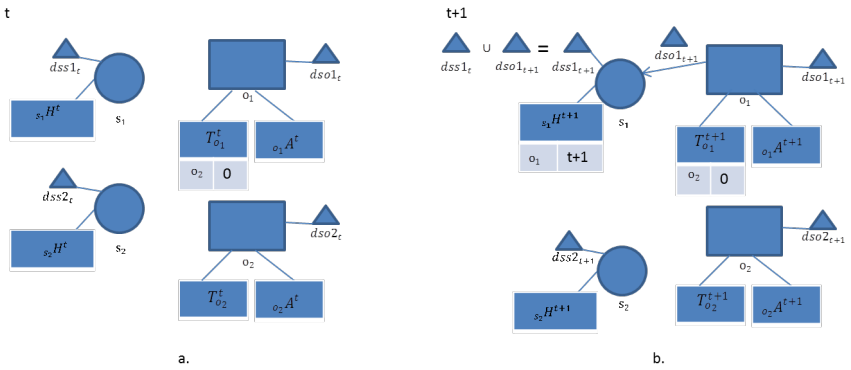

Fig. 1. Initial situation at instant of time t ($a.$) and read access at instant of time $t+1$ ($b.$) [1]

the subject adds the currently accessed object with the current time to its Read Access History.

Figure 1 ($b.$) shows the instant of time $t+1$, where the subject s_1 performs a read access operation to object o_1. As part of the read access the dataset of o_1 is added to the dataset of s_1 and a tuple consisting of the instance of time $t+1$ and o_1 is added to the Read Access History of s_1.[1]

If a subject performs a write access to an object, it writes its dataset into the dataset of the object. Additionally the Read Access History of the subject is merged with the Dataset Actuality of the object in a way that the Dataset Actuality of the object contains the entries of both histories afterwards. The entry with the more recent instance of access time is selected in case of a collision.

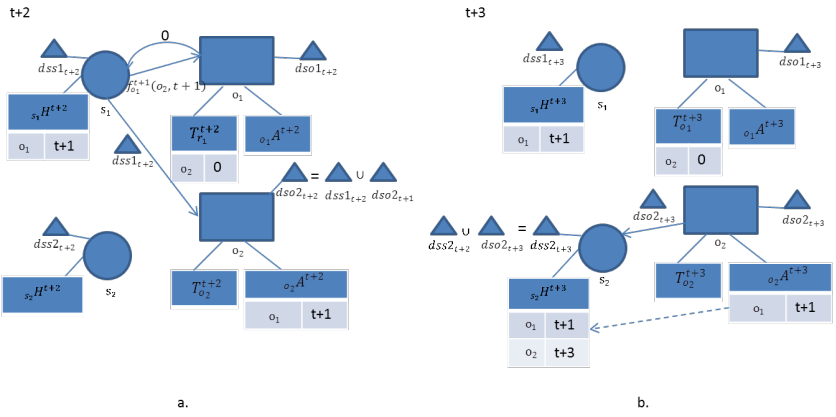

Fig. 2. Write access at instant of time $t+2$ ($a.$) and read access at instant of time $t+3$ ($b.$) [1]

Before a subject performs a write access, it invokes the Conflict Function of all the objects enlisted in its own Read Access History. The write access is denied, if one of the objects signals a conflict with the intended write access.

Figure 2 ($a.$) shows the instant of time $t+2$, where the subjects s_1 performs a write access to object o_2. Before it writes its dataset to o_2, s_1 has to invoke all the Conflict Functions of the objects enlisted in its own Read Access History. In addition to the merging of the dataset of s_1 into the dataset of o_2, a merging of the Read Access History of s_1 with the Dataset Actuality of o_2 is performed and written to the Dataset Actuality of o_2.[1]

Figure 2 ($b.$) shows the instant of time $t+3$, where subject s_2 performs a read access to o_2. As part of the read access, a merging of the Dataset Actuality of o_2 with the Read Access History of s_2 is performed and written to the Read Access History of s_2. After the Read Access History of s_2 is refreshed with a tuple of o_2 and the instant of time of the current read access, it contains the complete information about the sources and actuality of the data contained by the dataset dss2 of s_2.[1]

5 An agent based extension of the Limes Security Model

The Limes Security Model is introduced for pure subject/object relationships. In the use cases considered by the Limes Security Model objects only stay in conflict with objects and do not stay in conflict with a subject. In such scenarios the information flow between objects has to be controlled, which can be done by controlling the write accesses of the subjects. Agents in the sense of [2] are not taken into account. Agents implement services and can be accessed like objects and on the other hand may request services and act like subjects. Because of this ability objects and agents can stay in conflict with an agent and an agent can stay in conflict with other objects and agents. This implies that not only the write accesses of an agent have to be taken into account, but also the read accesses.

An agent combines the characteristics of both a subject and an object.

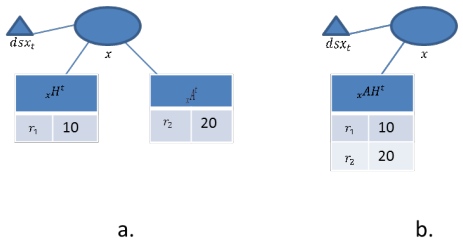

Fig. 3. Agent with Dataset Actuality (if considered to act like an object) and Read Access History (if considered to act like a subject) ($a.$) and with Dataset History ($b.$)

.

Definition 4. *Let the set of all agents SA be the set of all subjects that are objects, i.e. $SA := O_t \cap S$*

This means that an agent owns not only its dataset, but also a Read Access History if it acts like a subject and a Conflict Function and a Dataset Actuality in the case it is accessed like an object (shown for the agent x in Figure 3 $(a.)$). Because each agent has only one dataset, the Read Access History and the Dataset Actuality have to contain the same information. This is a prerequisite if both contain the complete information about the sources and actuality of the data contained by the agents dataset. If the agent acts as a subject, it has to synchronize its Read Access History with its Dataset Actuality and vice versa in case the agent is accessed like an object. Therefore it is obvious to define one history, the Dataset History. It is used as the Read Access History if the agent acts like a subject and as the Dataset Actuality in the case, where it is accessed like an object (shown for the agent x in Figure 3 $(b.)$). The Dataset History contains the complete information about the sources and actuality of the data contained by the dataset of the agent.

Definition 5 (Dataset History of an agent).
For each instant of time $t \in \mathbb{N}$ and each agent $x \in SA$ let ${}_xAH^t$ with the property

$$\forall\, j \in O_t :\ \forall\, t_1, t_2 \in \mathbb{N} : \big((j, t_1) \in {}_xAH^t \wedge (j, t_2) \in {}_xAH^t\big) \implies t_1 = t_2$$

be the Dataset History of the agent x. The Dataset History is the set of tuples (j, t') of the object or agent j that have been read accessed by the agent x in the past, or where the dataset has been read and has been written to x afterwards and the instant of time t' of the individual read access to j.

If $(j, t') \in {}_xAH^t$ then t' is the instant of time, where the last read access to object or agent j has been performed and where the read dataset has been written to x afterwards.

In addition to the Dataset History each agent owns a Dataset History Function as defined by Definition 6.

Definition 6 (Dataset History Function of an agent). *For each instant of time $t \in \mathbb{N}$ and each agent $x \in SA$ and all objects and agents $j \in O_t$ the value $t' = {}_xAH^t$ is the instant of time, where the read access to object j has been performed and where the read dataset has been added to the dataset of x later. The function $d_x^t : O_t \to \mathbb{Z}$ is called the Dataset History Function of x, if d_x^t has the following property:*

$$d_x^t(j) = \begin{cases} t', & \text{if } (j, t') \in {}_xAH^t, \\ -1, & \text{otherwise.} \end{cases} \tag{3}$$

If x is accessed like an object $d_x^t(j)$ is also termed as $a_x^t(j)$. If x acts like a subject $d_x^t(j)$ is also termed as $h_x^t(j)$.

5.1 Access limitations of an agent

Because of its object behavior, an agent can remain in conflict with different objects or agents. Before an agent is granted read access to an object or agent it has to be clarified if the object (or agent) itself or one of the objects (or agents) in the Dataset Actuality of the accessed object (or the Dataset History of the accessed agent) is in conflict with the accessing agent.

Definition 7 defines in 4 the condition under which a read access is granted and additionally the sanitization of the Dataset History of the accessing agent.

Definition 7 (Read access of an agent). *For each instant of time $t \in \mathbb{N}$, all objects $i \in O_t$, all agents $x \in SA$ and for all $r \in O_t$ with $a_i^t(r) \geq 0$ a read access from x to i is granted at instant of time t if and only if:*

$$(f_r^t(x, a_i^t(r)) \leq 0) \wedge (f_i^t(x, t) \leq 0) \tag{4}$$

If x performs a read accesses to i and (4) is fulfilled then let $D \subseteq O_t \times \mathbb{N}$ with

$$D := \{(i, d_x^t(i)) \mid d_x^t(i) \geq 0\} \cup \{(r, d_x^t(r)) \mid d_x^t(r) \geq 0 \wedge a_i^t(r) > d_x^t(r)\}$$

be the set of tuples in ${}_xAH^t$ that have to be removed from ${}_xAH^t$ because x receives from i as part of its dataset more recent data from the object r than x already owns in its own dataset. Also, let $M \subseteq O_t \times \mathbb{N}$ with

$$M := \{(i, t)\} \cup \{(r, a_i^t(r)) \mid \big(d_x^t(r) \geq 0 \wedge a_i^t(r) > d_x^t(r)\big) \vee d_x^t(r) = -1\}$$

be the set of tuples of objects r, where x receives from i as part of its dataset more recent data from the object r than x already owns in its own dataset and the instances of access times, when these objects have been read accessed, and when i has been accessed with the current instant of time t. If x performs a read access to i at instant of time t, then ${}_xAH^{t+1}$ is defined as

$${}_xAH^{t+1} := ({}_xAH^t \setminus D) \cup M.$$

Definition 8 defines in 5 the condition, under which a write access is granted and additionally the sanitization of the Dataset History of the accessing agent and of the Dataset Actuality of the accessed object.

Definition 8 (Write access of an agent). *For each instant of time $t \in \mathbb{N}$, all objects $r \in O_t$, all agents $x \in SA$ and for all $i \in O_t$ with $d_x^t(i) \geq 0$ a write access from x to r is granted if and only if:*

$$(f_i^t(r, d_x^t(i)) \leq 0) \wedge (f_x^t(r, t) \leq 0) \tag{5}$$

If x performs a write access to r and (5) is fulfilled then ${}_xAH^{t+1}$ is defined as

$${}_xAH^{t+1} := {}_xAH^t \setminus \{(i, d_x^t(i)) \mid d_x^t(i) \geq 0 \wedge f_i^t(r, d_x^t(i)) < 0\}.$$

Let $D \subseteq O_t \times \mathbb{N}$ with

$$D := \{(x, a_r^t(x)) | a_r^t(x) \geq 0\} \cup \{(i, a_r^t(i)) \mid d_x^t(i) \geq 0 \wedge a_r^t(i) \geq 0 \wedge d_x^t(i) > a_r^t(i)\}$$

be the set of tuples in ${}_rA^t$ that have to be removed from ${}_rA^t$, because r receives from x as part of the dataset from x more recent data from object i than r already possesses in its own dataset. Also, let $M \subseteq O_t \times \mathbb{N}$ with

$$M := \{(i, d_x^t(i)) \mid d_x^t(i) \geq 0 \wedge a_r^t(i) \geq 0 \wedge d_x^t(i) > a_r^t(i)\} \cup \{(i, d_x^t(i)) \mid d_x^t(i) \geq 0 \wedge a_r^t(i) < 0\} \cup \{(x, t)\}$$

be the set of tuples of objects i, where r receives as part of the dataset from x more recent data from the object i than r already owns in its own dataset and the instances of access times, when these objects have been read accessed. If x performs a write access to r at instant of time t and (5) is fulfilled then ${}_rA^{t+1}$ is defined as

$${}_rA^{t+1} := ({}_rA^t \setminus D) \cup M.$$

5.2 Illegal Information Flow

Similarly to the Limes of a subject, defined in [1], the Limes of an agent describes the borderline between the objects (and agents), where an agent is allowed to perform a read or write access, and those objects (and agents), where the agent is not allowed to perform a read or write access. The Limes of an agent is the set of objects (and agents) that the agent is currently not allowed to read or write access because of a conflict of interest.

Definition 9 (Limes For Write Access of an agent). *For each instant of time $t \in \mathbb{N}$ and all agents $x \in SA$ let the Limes For Write Access ${}_xLW^t \subseteq O_t$ of x at instant of time t be defined as*

$${}_xLW^t := \{j \in O_t \mid \exists\ i \in O_t :\ (d_x^t(i) \geq 0 \wedge f_i^t(j, d_x^t(i)) > 0) \vee f_x^t(j, t) > 0\}.$$

Definition 10 (Limes For Read Access of an agent). *For each instant of time $t \in \mathbb{N}$ and all agents $x \in SA$ let the Limes For Read Access ${}_xLR^t \subseteq O_t$ of x at instant of time t be defined as*

$${}_xLR^t := \{j \in O_t \mid \exists\ i \in O_t :\ (a_j^t(i) \geq 0 \wedge f_i^t(x, a_j^t(i)) > 0) \vee f_j^t(x, t) > 0\}.$$

Definition 11 (Illegal Information flow). *For each instant of time $t \in \mathbb{N}$ and each agent $x \in SA$ and all objects or agents $j \in {}_xLW^t$ and all objects or agents $i \in {}_xLR^t$ an illegal information flow takes place, if x performs a write access to j or x performs a read access to i at instant of time t.*

Theorem 1. *For each instant of time $t \in \mathbb{N}$ and all agents $x \in SA$ and all objects or agents $i, j \in O_t$ implementing their Conflict Function according to Definition 2, there will be no illegal information flow between i, j or x, if x only performs a read or write access according to Definitions 7 and 8, and if x is only read or write accessed according to Definitions 7 and 8.*

Proof. Assumption 1: There has been an illegal information flow from object or agent i to object or agent j because of a read access to i and a following write access to j by agent x, although the objects and agents have implemented their Conflict Function according to Definition 2 and the read and write accesses of x have been performed according to Definitions 7 and 8.

Conforming to Definition 11 an illegal information flow from i to j happens, when x performs a write access to j in case $j \in {}_xLW^t$, which is true after the read access of x to i or in general after a read access to i, where the read dataset has been written to x later and if $f_i^t(j, d_x^t(i)) > 0$.

If $d_x^t(i) \geq 0 \wedge f_i^t(j, d_x^t(i)) > 0$. From $d_x^t(i) \geq 0$ follows that $d_x^t(i)$ is the instant of time of the last read access of x to i or in general the last read access to i, where the read dataset has been written to x later.

From Definition 2 and $f_i^t(j, d_x^t(i)) > 0$ follows that i is in conflict with j. Definition 11 specifies that a write access is only allowed if for every $i \in O_t$ with $d_x^t(i) \geq 0$ the Conflict Function delivers a result $f_i^t(j, d_x^t(i)) \leq 0$. This contradicts the assumptions that the objects and agents have implemented correctly their Conflict Function according to Definition 2 and the read and write accesses have been performed according to Definitions 7 and 8.

Assumption 2: There has been an illegal information flow from agent x to object or agent j because of a write access to j by agent x although the objects and agents have implemented their Conflict Function according to Definition 2 and the write access of x has been done according to Definition 8.

By Definition 11 an illegal information flow from x to j happened, because of the write access of x to j if $j \in {}_xLW^t$, which is the case if $f_x^t(j, t) > 0$. It follows that x is in conflict with j. Definition 11 defines that a write access is only allowed if the Conflict Function of x delivers a result $f_x^t(j, t) \leq 0$. This contradicts the assumptions that x has implemented its Conflict Function according to Definition 2 and the write access has been executed according to Definition 8.

Assumption 3: There has been an illegal information flow from object or agent i to agent x because of a read access to i by agent x although the objects and agents have implemented their Conflict Function according to Definition 2 and the read access of x has been executed according to Definition 7.

By Definition 11 an illegal information flow from i to x happened, because of the read access of x to i if $i \in {}_xLR^t$, which is the case if $f_i^t(x, t) > 0$. It follows that i is in conflict with x. Definition 11 defines that a read access is only allowed if the Conflict Function of i delivers a result $f_i^t(x, t) \leq 0$. This contradicts the assumptions that i has implemented its Conflict Function according to Definition 2 and the read access has been executed according to Definition 7.

6 Conclusion and Future Works

Service-oriented architectures have gained in importance by the implementation of business processes as a dynamic and loosely coupled system of services. In [9] Pratt and Zeckhausen discuss a problem that is based on an asymmetric relationship because of an informational advantage of the service compared to the

client. The problem lies in the uncertainty of the client, not knowing whether the agent might misuse his data. To guarantee an approved protection of the client data and to prevent an illegal information flow, an approved security policy model has to be chosen that is suitable for the considered use case.

The Limes Security model introduced in [1] acts on the assumption that a conflict of interest is not necessarily symmetric, not necessarily reflexive and not necessarily transitive. It is introduced for pure subject/object relationships, where agents are not considered. An agent combines the characteristics of both a subject and an object. In the current paper, the Limes Security model has been extended to prohibit an illegal information flow in an agent based service-oriented architecture. In any kind of access the complete dataset of an object, subject or agent is read or written, which probably is a limitation for some possible use cases. For the future work this point has to be dicussed in-depth. Additionaly a prototype implementation of the security policy model and a performance evaluation needs to be conducted.

7 Acknowledgement

The author thanks Prof. Dr. Fuss for his openness to discussion on mathematical questions and Prof. Dr. Grimm and Dipl-Inf. Dieter Kessler for there support in the preparation of this paper.

References

1. Hermann, Eckehard, *The Limes Security Model for Information Flow Control*, accepted at FARES Workshop of the Sixth International Conference on Availability, Reliability and Security (ARES 2011), Vienna, Austria, Aug 22-26, 2011.
2. Booth, David, Haas, Hugo, McCabe, Francis, Newcomer, Eric, Champion, Michael, Ferris, Chris and Orchard, David, *Web Services Architecture*, W3C Working Group Note, 11 February 2004, http://www.w3.org/TR/ws-arch/, 2004.
3. Brewer, D. F. C. and Nash, M. J., *The Chinese Wall Security Policy*, in IEEE Symposium on Security and Privacy, Oakland, pp. 206–214, 1989.
4. Lin, T. Y. *Chinese Wall Security Policy-An Aggressive Model*, Proceedings of the Fifth Aerospace Computer Security Application Conference, December 4-8, pp. 286-293, 1989.
5. Bell, D. and LaPadula, L., *Secure Computer Systems: Mathematical Foundations*, MITRE Corporation, Bedford, MA, Technical Report MTR-2547, Vol. I., 1973
6. Clark, D. and Wilson, D., *A Comparison of Commercial and Military Security Policies*, in IEEE Symposium on Security and Privacy, pp. 184 -194, 1987.
7. Loock M and Eloff JHP, *A new Access Control model based on the Chinese Wall Security Policy Model*, in Proceedings of the ISSA 2005 New Knowledge Today Conference, Information Security South Africa (ISSA), pp 1-10 (CD), 2005.
8. S. Burbeck, *The Tao of E-Business Services*, IBM developerWorks, 2000; www-128.ibm.com/developerworks/libraryws-tao .
9. Pratt, John W., Zeckhausen, Richard J., *Principals and Agents: The Structure of Business*, Harvard Business School Press, Boston, 1985

10. Ricci, Alessandro, Buda, Claudio, Zaghini, Nicola, *An Agent-Oriented Programming Model for SOA & Web Services*, 5th IEEE International Conference on Industrial Informatics, Vienna, 2007.
11. Yu-Cheng Hsiao and Gwan-Hwan Hwang, *Implementing the Chinese Wall Security Model in Workflow Management Systems*, in Proceedings of the International Symposium on Parallel and Distributed Processing with Applications (ISPA '10). IEEE Computer Society, Washington, DC, USA, 574-581, 2010.
12. Patrick C. K. Hung and Guang-Sha Qiu, *Implementing Conflict of Interest Assertions for Web Services Matchmaking Process*, 2003 IEEE International Conference on E-Commerce Technology (CEC'03), Newport Beach, California, USA, 2003
13. Jana Debasish, Bihari Bhaumik Bijan, *Security Model of Service Oriented Computational Grids*, India Conference, 2006 Annual IEEE, New Delhi, 15-17 Sept. 2006.
14. Ruoyu Wu, Gail-Joon Ahn, Hongxin Hu, Mukesh Singhal, *Information flow control in cloud computing*, 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), Chicago, 2010.