A Classification of Interdomain Actions

Winfried E. Kühnhauser
GMD National Research Center For Information Technology, Sankt Augustin, Germany

This paper contributes to the recent discussion on unmet information security challenges for operating system designers. It focuses on the problem that in order to meet these challenges operating systems must be capable of supporting a multitude of information domains, each domain defined by its own and individual security policy. In such multi-domain systems, the inter-operability between different information domains constitutes a major problem. While the security policies of the system control the interactions within their domains, it is an unsolved problem how interactions between different domains can be made secure.

In order to provide a precise foundation for the discussion of secure interdomain actions as well as for the development of concepts for their implementation, the paper proposes a classification of interdomain actions that clearly identifies two major types of interdomain actions: interactions that cause conflicts between the involved security policies and interactions for which none of the involved policies can provide any security rule. The paper concludes that in order to support multiple information domains, operating systems must be capable of classifying interdomain actions, and they must support new types of interdomain security policies that mediate security conflicts in interdomain actions and complete the set of security rules for interdomain actions. The paper concludes with a discussion of the computational complexity of interdomain action classification.

Keywords: security policy, multipolicy system, information domain, policy domain.

1 Introduction

In their recent paper on unmet information security challenges [FM98], Edward Feustel and Terry Mayfield identified three major areas that have to be addressed by operating system designers in order to meet the information security challenges imposed by the new Department of Defense Goal Security Architecture (DGSA) [Def96]. Feustel and Mayfield argued that

- operating systems must be capable of supporting a wide range of security policies simultaneously
- operating systems must be capable of submitting off-the-shelf application software to a protected environment where a security policy is enforced on the application
- operating systems must allow distributed processing, and they must be sufficiently protected to allow connectivity via public networks.

A key to multiple security policies is the concept of information domains. The basic idea is similar to that of confidence domains introduced in [Küh94]: an information domain is a set of entities (such as users, processes, files, mailboxes) together with a single encapsulating security policy. Multi-Domain systems will have as many information domains as there are security policies.
While information domains provide a clear concept to describe the relationship between a security policy and the entities it controls, information domains are a threat to interoperability; they tend to establish autonomous islands that are protected by their security policy, and it is unclear how trips between the islands can be made and managed with respect to security.

The major obstacle to interoperability between information domains is the isolation enforced by its security policy. For any well-defined domain it is precisely defined whether an entity belongs to the domain or not, and the domain’s security policy will contain precisely all the rules that control the interactions of those entities in its domain. Now consider an interaction between entities that belong to different domains. Even in the most simple case when a subject from some domain $P$ accesses an object in another domain $Q$, at least two policies are involved.

None of the two policies will be able to provide a rule for this access: policy $P$ will not know the object and its security attributes within domain $Q$, and neither will policy $Q$ know the subject and its security attributes within domain $P$.

In a second example, domain $P$ overlaps with domain $Q$, and subject and object are located within the overlapping section (this is one way to implement Feustel’s and Mayfield’s interdomain transfer objects). In this case, both security policies know both entities and might come up with two and possibly conflicting rules.

This paper is a first step to meet the challenge of interdomain actions. In order to gain some insight in the nature of interdomain actions, section two discusses a detailed example of a large-scale multi-domained distributed system. The focus of the discussion will be on interdomain authentication; within the limits of a journal paper, this will help us to conduct the discussion on a sufficiently detailed level. In order to precisely identify the security aspects related to interdomain actions, section three then abstracts from the example and develops a general classification of interdomain actions. Based on this classification, the last section outlines a concept for their implementation.
2 A Multi-Domain Scenario

Our example scenario is a large corporation developing sophisticated technology such as satellite navigation systems, cockpit air conditioning systems or electronic engine management systems. Our corporation owns sites in every country where it sells its products. Local branches of the development department customize products to country-specific needs such as language, technical standards or laws; local branches of the marketing department observe country-specific markets and maintain relationships to clients. Examples for centralized departments are research, development and purchasing.

Because many branches and departments will have their own unique information security requirements, any global security policy that is effective within the entire corporation will be very general and will not be able to take into account the individual security requirements of any particular branch, department, or project. To that end, subunits of the corporation usually define their own individual security policies, which can be refinements or "stronger" versions of the global security policy or, in special cases, be totally disjunct from it. Refinements of the global security policy will result in a hierarchy of security policies, where the global security policy forms the outer shell, and the innermost shell protects the most secret research projects, using for example a mandatory access control policy and cryptosystems that are totally incompatible with the other shells of the policy hierarchy (figure 3).

Apart from hierarchical domain structures where security requirements become stronger when moving to the inner shells, there also examples in which the corporation's global security policy will be weakened. Consider for example the public relations department that manages the presentation of the corporation in the Web; in contrast to every other server of the corporation's distributed system, the Web servers must be accessible for any foreign system (e.g. by locating them outside a firewall (figure 4)).

In order to improve cooperation between its departments the corporation encourages its employees to transfer their place of work temporarily to other departments. As an example,
experts from the research department may move for a few weeks to the product development department in order to support a fast dissemination and exploitation of their work. A sales manager from the marketing department may move to the development department to become familiar with a new product, and experts from central product development may move to the corporation’s country-specific sites to help their colleagues with the customization of a new product.

The temporary movement of employees to other departments and, consequently, to other security policy domains requires a security concept that explicitly handles the cooperation between the domains. As an example, the following security requirement grants an employee who has moved for some weeks to a new department access to his familiar working environment as well as it allows him to cooperate with his colleagues in his temporary department.

**Security Requirement Example:** Any person who temporarily transfers her/his place of work into another security domain will keep all access rights from her/his home domain and will additionally obtain the rights of an employee of her/his temporary assignment.

### 2.1 Security Policies

Before proceeding any further, we want to establish a common understanding of the term security policy. In this paper, the term "security policy" denotes a systematical transformation of a set of security requirements into a strategy (a set of rules, an algorithm) that implements the security requirements. Depending on the abstraction level, security policies are represented by (informal) human language, (formal) security models, or executable code units.

There is a general distinction between a *corporate* security policy and a *technical* security policy. While a corporate security policy covers all aspects concerning the management, the protection and distribution of sensitive information within an organization (including aspects such as delegation of responsibility to humans or controlling human access to rooms), a technical security policy focuses on those issues that are relevant within a computer system. Within this paper we focus on the technical security policies.

A large class of security policies – policies aiming at integrity and confidentiality – rests on
two pillars: authentication and authorization. Authentication is concerned with the correct identification of principals: entities that have names within a computer system, such as users, processes, or files. Authorization is concerned with rules that govern access operations. The sum of all authentication rules is called the \textit{authentication scheme} of a security policy; the sum of all access rules is called the \textit{authorization scheme}. The authentication of principals is a prerequisite for any authorization scheme: to apply the rule "Joe might read file X" the identity of Joe and the file X must be established correctly.

Considering our example scenario, the different security policies of different domains will contain individual authentication and authorization schemes. Departments with lower security requirements for example might use a simple password based authentication scheme, discretionary access control and a simple and fast encryption function to protect network communication. Other departments with stronger security requirements will use more sophisticated schemes such as chipcard based user authentication, an authentication scheme between clients and servers where clients as well as the servers mutually authenticate themselves, a mandatory access control policy and public-key encrypted communication and data storage.

And here, in the incompatibility of the authentication and authorization schemes we meet a major reason for the interoperability problems of interdomain actions: the identity of users from one domain with a password based authentication may not be accepted within another domain with a stronger chipcard authentication scheme, the authorization schemes may be of different type (discretionary vs. mandatory), and cryptosystems may be incompatible due to the use of different key types (symmetric vs. asymmetric), key lengths, or encryption functions.

2.2 Authentication

Now let us take a closer look at the mutual authentication of user and server within this scenario. Active parties in an authentication are

- the user,
- the authentication authority of the user, implementing the authentication scheme (e.g. a Kerberos authentication server as described in [MNSS88, Koh91])
- the server
- the authentication authority of the server.

Within our scenario, these components may be distributed in various constellations among the security domains. For the discussion of these constellations we assume that in order to protect integrity or confidentiality, communication between user\textsuperscript{1} and server is encrypted. We also assume that, as it is a common practise, distribution of the necessary keys is part of the authentication protocol. To restrict the number of different constellations in our example we finally assume that in contrast to users the server is not mobile.

Now consider the different constellations a mobile user may encounter.

\textsuperscript{1}respectively the process acting in behalf of the user
1. In the first constellation the user works on his usual workstation located within his home security domain $Q$ and uses a server within the same domain.

![Figure 5: Authentication and service request within one security domain](image)

Authentication of both user and server is done by the authentication authority of their home domain. Because both parties trust this authority they also trust its statements about the identity of their communication partners as well as in the proper construction of the keys. Thus both parties may now communicate: they believe in the identity of their communication partner, have proper encryption keys and use the same cryptosystem.

In the second and third constellation a member of department $Q$ temporarily moved to another department $P$. However, during this time he needs access to the servers of his home domain as well as to the servers in the security domain of his temporary assignment.

![Figure 6: Authentication variants and same service request across security domains](image)

2. In this constellation authentication of the user is done by the authentication authority of his temporary department (alternative (a) in figure 6); we observe the following
properties here:

- The user has to use a foreign authentication scheme; for example, he may be forced to use a password based scheme instead of the chipcard based scheme he is used to. In particular, he is forced to trust a foreign authentication authority that it keeps the secrets used during the authentication. If the scheme is weaker than that of the mobile user’s home domain (due to flaws in the protocol, unencrypted password exchange, malicious security administrator etc.), it may become possible for users in department $P$ to masquerade as our user and thus get access to the more security-critical servers within domain $Q$.

- For the user to be able to access the server within his home domain, this server has to accept the authentication authority of domain $P$; a weaker scheme in $P$ thus is a security threat to domain $Q$.

- Due to different security requirements in both domains the cryptosystems in $P$ and $Q$ might be incompatible (symmetric/asymmetric encryption, different hash functions, different key lengths). In this case, the authentication authority within $P$ must be put in the position to issue keys for interaction with domain $Q$.

3. In this constellation authentication of the user is done by the authentication authority of his home department (alternative (b) in figure 6); we observe the following properties here:

- Within domain $P$, the necessary infrastructure for the authentication scheme of $Q$ must be present; for example if the scheme in $Q$ uses chipcards, the workstation of our user in $P$ must have a chipcard reader.

- For the user to be able to access the servers within his temporary domain, these servers have to accept the authentication authority of domain $Q$; a weaker scheme in $Q$ thus is a security threat to domain $P$.

- The cryptosystems in $P$ and $Q$ might be incompatible (see above); the authentication authority within $Q$ must be put in the position to issue keys for servers within domain $P$.

- If the authentication scheme in $Q$ uses passwords, the user’s password is disclosed at least to his workstation within $P$, resulting in a security risk for $Q$.

2.3 Conclusions of the Example Scenario

The discussion of our example indicates that authentication in multi-domain systems requires strategies that precisely define the conditions under which users are authenticated by the authentication authorities. Considering that authentication is just one aspect of information security (the authorization scheme is another), the need for interdomain security policies is obvious. These policies go far beyond the scope of simple security mechanisms; in the same way as security policies regulate interactions within the borders of their domain, interdomain security policies define the security properties of interdomain actions.

The following section is a first step to approach the notion of interdomain security policies. It discusses the nature of interdomain actions and aims at a classification that will help to identify
the role of interdomain security policies more precisely. We will show that this classification is both, unique and complete.

3 Classification of Interdomain Actions

This section aims at a deeper understanding of the nature of interdomain actions. For the sake of conciseness, we will restrict the scope of the discussion to access control policies.

We then may use Lampson’s traditional approach to access control modelling in which any interaction between the entities of a system is modelled as a subject \( s \) accessing an object \( o \) via an access operation \( (s, o) \). We write \( P(s, o) \) when we apply policy \( P \) to a specific access \((s, o)\); \( P \) is an access control policy, so \( P(s, o) \) is of type \{\textit{granted}, \textit{rejected}\}. We use \( \text{Dom}_P \) to denote the domain belonging to \( P \) which denotes all entities (subjects and objects) that are submitted to policy \( P \). For any access \((s, o)\), a policy will contain an access rule if and only if \( s, o \in \text{Dom}_P \). \( \Pi_e = \{ P | e \in \text{Dom}_P \} \) denotes the set of security policies that have entity \( e \) within their domain. Last but not least we use \(|M|\) as the usual denotation for the cardinality of some set \( M \).

In any multi-domain scenario with a set of security policies \( P \in I \), where \( I \) is a finite set of indices, any access \((s, o)\) with \( s, o \in \bigcup_{i \in I} \text{Dom}_P \) belongs to one of the following three classes.

\begin{itemize}
  \item \textbf{Class 1:} \(|\Pi_s| = |\Pi_o| = 1 \land \Pi_s = \Pi_o|

  This access class is characterized by the situation that subject and object are members of the same domain and not member of any other domain.

  \end{itemize}

\begin{figure}[h]
\centering
\includegraphics[width=0.2\textwidth]{access_class1.png}
\caption{An example for access class 1}
\end{figure}

Here, \((s, o)\) is no interdomain action, and consequently, policy \( P \) is both capable and authorized to make the access decision.

\begin{itemize}
  \item \textbf{Class 2:} \(|\Pi_s \cap \Pi_o| = 0|

  This access class is characterized by the situation that no security policy exists that has both, subject and object, in its domain. Especially, there is no security policy that is capable of providing a rule for this particular access. Nevertheless, because \( s, o \in \bigcup_{i \in I} \text{Dom}_P \), there are security policies that encompass one of the entities within their domain; here, we might further distinguish between

  \begin{enumerate}
    \item \(|\Pi_s| = 1 \land |\Pi_o| = 1|

  \end{enumerate}

\end{itemize}
Figure 8: An example for access class 2a

where both entities are member of precisely one domain (figure 8).

(b) $\exists e \in \{s, o\}$ with $|\Pi_e| > 1$,

where at least one of the entities is a member of more than one domain; figure 9 gives an example with four domains.

Figure 9: An example for access class 2b with four domains

Class 3: $|\Pi_s \cap \Pi_o| \geq 1 \land \exists e \in \{s, o\}$ with $|\Pi_e| > 1$

This access class is characterized by the situation that on the one hand there is (at least) one policy that might provide a rule for the access; nevertheless, on the other hand (at least) one of the entities is a member of more than one domain. Further we distinguish between the following cases.

(a) $|\Pi_s \cap \Pi_o| = 1$

In this case there exists exactly one policy providing a rule for the particular access; however this rule may not be applied alone because one of the entities ($o$ in figure 10) belongs to a second domain.

Figure 10: An example for access class 3a

(b) $|\Pi_s \cap \Pi_o| > 1$
In this case there exists more than one policy that might yield a rule for the particular access. Additionally there might be more policies that encompass one of the entities in their domain (R in figure 11).

![Diagram of domains and access]

Figure 11: An example for access class 3b

### 3.1 Uniqueness and Completeness

This classification of interdomain actions is both unique and complete: any access \((s, o)\) with \(s, o \in \bigcup_{i \in I} \text{Dom}_i\) belongs to exactly one class. This is important because we will use the individual classes to identify the roles of interdomain security policies.

#### Proof of Uniqueness

In order to prove the uniqueness of our classification, we assume for each class that its characteristic class property holds and deduce a contradiction to the remaining two class properties.

(a) Let us assume a class 1 access, so that \(|\Pi_s| = |\Pi_o| = 1 \land \Pi_s = \Pi_o\) holds.

   (a1) The contradiction to class 2 property \(|\Pi_s \cap \Pi_o| = 0\) follows directly from \(\Pi_s = \Pi_o\).

   (a2) Class property 3 requires an \(e \in \{s, o\}\) with \(|\Pi_e| > 1\); because \(|\Pi_s| = |\Pi_o| = 1\), there cannot be such an \(e\).

(b) Let us assume a class 2 access, so that \(|\Pi_s \cap \Pi_o| = 0\) holds.

   (b1) Because each \(\Pi\) is non-empty, the contradiction to class 1 property \(\Pi_s = \Pi_o\) follows directly from the assumption.

   (b2) The contradiction follows directly from \(|\Pi_s \cap \Pi_o| \geq 1\).

(c) Let us assume a class 3 access, so that \(|\Pi_s \cap \Pi_o| \geq 1 \land \exists e \in \{s, o\} : |\Pi_e| > 1\) holds.

   (c1) The existence of an \(e\), \(|\Pi_e| > 1\) is a contradiction to \(|\Pi_s| = |\Pi_o| = 1\).

   (c2) The contradiction follows directly from \(|\Pi_s \cap \Pi_o| = 0\).

\(\square\)
Proof of Completeness

In order to prove the completeness of our classification we will distinguish between the cases

(a) \(|\Pi_s \cap \Pi_o| = 0\)
(b) \(|\Pi_s \cap \Pi_o| = 1\)
(c) \(|\Pi_s \cap \Pi_o| > 1\)

Each of these cases is covered by our classification: each access of type (a) belongs to class 2; in case (b), if \(s\) and \(o\) belong to exactly one domain (that is, \(|\Pi_s| = |\Pi_o| = 1\)), the access belongs to class 1; else an \(e\) exists with \(e \in \{s, o\} : |\Pi_e| > 1\), and the access belongs to class 3. In case (c), \(|\Pi_s| > 1\) and \(|\Pi_o| > 1\) holds; the \(e\) requested in access class 3 thus exists.

\(\square\)

Thus we now have a classification of interdomain actions in multi-domain scenarios that it is both complete and unique. The next section will show how this classification identifies the roles of interdomain security policies.

### 3.2 Class-One Interdomain Actions

Let us briefly consider what we have achieved so far. The scenario we are discussing has a multitude of security policies. Each security policy has its own area of responsibility, called its domain. Global interoperability in multi-domained systems requires interdomain actions; these actions are not covered by the security policies but must be controlled by additional interdomain security policies. In order to more deeply understand the role of interdomain security policies we provided a classification that identified three types (or classes) of interdomain actions. Class-one actions can be handled by the existing security policies. Class two characterizes interdomain actions for which none of the existing security policies is in a position to provide a rule (see also the example in the introduction); interdomain actions within this class are policy-free. Class three characterizes interdomain actions that involve more than one policy, giving cause to different types of conflicts. The following definitions 1-3 precisely define the meaning of policy-free and conflicting interdomain actions.

Let \(I\) be a finite index set, \(\{P_i\}_{i \in I}\) the set of security policies of a multipolicy system and \(s, o \in \bigcup_{i \in I} Dom P_i\).

**Definition 1**

An interaction \((s, o)\) is called policy-free if and only if there is no security policy that yields a rule for this interaction:

\[|\Pi_s \cap \Pi_o| = 0.\]
Definition 2
An interaction \((s, o)\) results in a \textit{domain conflict} if and only if at least one of the involved entities belongs to more than one policy domain:

\[ \exists e \in \{s, o\} : |\Pi_e| > 1. \]

Definition 3
An interaction \((s, o)\) results in a \textit{rule conflict} if and only if more than one security policy yields a rule for this interaction:

\[ |\Pi_s \cap \Pi_o| > 1. \]

Note that any rule conflict implies a domain conflict.

Now we are in the position to exactly state the necessary and sufficient conditions for inter-domain security policies.

Definition 4
An access \((s, o)\) is called \textit{interdomain policy free} if and only if it neither is policy-free nor causes a domain or rule conflict.

Corollary 5 supplies a more handy version of Definition 4.

Corollary 5
An access \((s, o)\) is interdomain policy free if and only if it is a class-one access.

Proof of Corollary 5
1. left to right: "\((s, o)\) is interdomain policy free \( \Rightarrow |\Pi_s| = |\Pi_o| = 1 \land \Pi_s = \Pi_o\)"

\((s, o)\) is interdomain policy free
\[ \Rightarrow_{\text{Def. 1-3}} \neg ( \exists e \in \{s, o\} : |\Pi_e| > 1 ) \land \neg ( |\Pi_s \cap \Pi_o| > 1 ) \land \neg ( |\Pi_s \cap \Pi_o| = 0 ) \]
\[ \Rightarrow \forall e \in \{s, o\} : |\Pi_e| \leq 1 \land |\Pi_s \cap \Pi_o| \leq 1 \land |\Pi_s \cap \Pi_o| \neq 0 \]
\[ \Rightarrow \forall e \in \{s, o\} : |\Pi_e| = 1 \text{ (any such } e \text{ is member of at least one domain)} \]
\[ \land |\Pi_s \cap \Pi_o| = 1 \]
\[ \Rightarrow |\Pi_s| = |\Pi_o| = 1 \land \Pi_s = \Pi_o \]

2. right to left: "\(|\Pi_s| = |\Pi_o| = 1 \land \Pi_s = \Pi_o \Rightarrow (s, o)\) is interdomain policy free"

\(\Pi_s = \Pi_o \Rightarrow \) not policy-free, no rule conflict
\[ |\Pi_s| = |\Pi_o| = 1 \Rightarrow \) no domain conflict.
3.3 Class-Two Interdomain Actions

For any class-two interdomain action we are stuck with the fact that none of the existing security policies yields a rule for the action. Although there might be more than one domain that contains either the subject or the object there is no domain that contains both entities (figures 8, 9).

Consequently, in order to restore well-defined information security properties of a system additional rules that implement interdomain security requirements have to be furnished. In other words, class-two accesses require an additional completeness policy that closes this hole.

3.4 Class-Three Interdomain Actions

For any class-three interdomain action we are stuck with the fact that at least one security policy yields a rule for the action, and at least one of the involved entities is a member of at least one other policy's domain (figures 10, 11). This accounts for different types of conflict. In case of a rule conflict, the conflict is caused by more than one policy yielding a rule for the access; in case of domain conflicts, an entity is member of more than one policy domain.

These conflicts cannot be mediated by the existing policies alone. Additional security requirements that regulate conflict resolution have to be furnished, resulting in a security policy implementing conflict mediation for interdomain actions.

4 Implementation

We have implemented interdomain actions as part of the security policy implementation within the Distributed Computing Environment (DCE) of the Open Systems Foundation (OSF). However, the essential parts of the implementation are platform-independent, so that our statements concerning the computational complexity are generally valid.

The essential parts of the interdomain action concept are

- a completeness policy \( T \)
- a conflict mediation policy \( F \)
- a classification function \( c \).

\( T \) and \( F \) are implemented in exactly the same paradigms and are enforced with exactly the same mechanisms as any regular security policy of a system. However, in contrast to any regular security policy, their domains encompass all domains of every regular security policy:

\[
\text{Dom}_T = \text{Dom}_F = \bigcup_{i \in I} \text{Dom}_{P_i}.
\]

The classification function \( c \) is of type \( S \times O \to \{P_i\}_{i \in I} \cup T \cup F \) (\( S, O \subset \bigcup_{i \in I} \text{Dom}_{P_i} \) are the
subject and object sets) and is defined by

\[ c(s, o) = \begin{cases} 
P_k & |\Pi_s| = |\Pi_o| = 1 \land \Pi_s = \Pi_o = \{P_k\} \text{ (class-one interaction)} \\
T & |\Pi_s \cap \Pi_o| = 0 \text{ (class-two interaction)} \\
F & |\Pi_s \cap \Pi_o| \geq 1 \land \exists e \in \{s, o\} : |\Pi_e| > 1 \text{ (class-three interaction)} 
\end{cases} \]

The classification function becomes part of the system’s (multi-policy) reference monitor that implements the total access mediation property \([Dep83]\). It substitutes the regular security policy call that is issued by the reference monitor on every reference (entity interaction). While class-one interactions proceed to the responsible security policy, class-two and class-three interactions are diverted to \(T\) respectively \(F\).

Any component of a reference monitor that is involved in implementing the total access mediation property is crucial to a system’s overall performance. The computational complexity of \(c\) depends on the complexity of determining the membership of an entity to all the system’s policy domains and on the complexity of set primitives such as computing set intersection or the cardinality of a set. Considering that while entity interactions occur frequently, security domains in general change rather rarely, our decision was to aim at an implementation with fast lookups and slow updates. As a consequence, updates of a policy domain run in \(O(n)\) (\(n\) is the number of entities joining or leaving a domain); classifying a particular interaction runs in \(O(1)\). A very simple implementation for the OSF DCE can be found at http://set.gmd.de/~kuhnhsr/mp.ps; this implementation omits reference counters and hash-keyed set implementation but still updates a policy domain in \(O(|I| \ast |\bigcup_{i \in I} Dom_{P_i}|)\).

5 Conclusions

In large organizations, many branches and departments have their own unique information security requirements. Mapping the structure of such organizations to a computer system results in a distributed system with a multitude of information security policies, each security policy having its individual and unique security domain. Supporting multiple security domains has been identified as a major challenge for operating systems.

In multi-domain systems, the interoperability between different domains constitutes a major problem. A classification of interdomain operations identified two major problem areas: interdomain actions that cause conflicts between the involved security policies and interdomain actions for which none of the involved policies provides any security rule.

The paper argued that in order to deal with these obstacles, new security requirements for interdomain actions must be furnished, and that based on these requirements, security policies dealing with conflicts and completeness in interdomain actions must be put in place and guard domain interactions at runtime.

Necessary and sufficient conditions for invoking these policies were identified. Checking these conditions becomes part of the reference monitor that implements the total access mediation property. A discussion of the computational complexity showed that this checking can be implemented efficiently in constant time.
References


