

Specifying and enforcing a multi-policy paradigm for high assurance multi-enclave systems

Lu'ay A. Wahsheh and Jim Alves-Foss *

Center for Secure and Dependable Systems, University of Idaho, P.O. Box 441008, Moscow, Idaho 83844-1008, USA
E-mail: {luay,jimaf}@uidaho.edu

Abstract. One fundamental key to successful implementation of secure high assurance computer systems is the design and implementation of security policies. For systems enforcing multiple concurrent policies, the design and implementation is a challenging and difficult task. To simplify this task, we present an Inter-Enclave Multi-Policy (IEMP) paradigm for information access of the Multiple Independent Levels of Security and Safety (MILS) approach to high assurance system design for security- and safety-critical multi-enclave systems. The IEMP paradigm manages multiple security policies (i.e., controls the conflicts and cooperation of policies of different enclaves) within heterogeneous systems. IEMPs are “policies about policies” that ensure the enforcement of end-to-end mandatory information flow security policies, where the management and evolution of policies can be separated from applications. Although the approach was initially designed for use in the MILS architecture, based on the concept of a separation kernel, it is applicable to a much broader range of architectures.

Keywords: Multi-policy, access control, information flow, policy management, conflicts, multi-enclave, MILS, high assurance

1. Introduction

As the use of computer systems becomes more commonly employed, managing their security becomes more complex. With the coexistence of different distributed environments communicating with one another and sharing resources, security is expressed using different types of policies that control information access. One fundamental key to successful implementation of secure high assurance computer systems is the design and implementation of security policies. These policies must specify the authorized transactions of the system and actions for unauthorized transactions, all in a form that is implementable. Implementing the enforcement of a policy is difficult and becomes very challenging when the system must enforce multiple policies.

1.1. What is policy?

In computer security, the term *policy* means different things to different people. Policies can be a set of rules to manage resources (e.g., actions based on a certain event(s)) or definite goals to determine present and future decisions. Compliance with policy is mandatory. Policies are different from guidelines, which are optional and recommended actions. Since they are not consistent, guidelines often violate systems' security. For some people, a policy should specify *what* data and resources need to be protected, *how* they will be protected, or *why* they need to be protected, while for others, it could be any combination of *what/how/why*. Broadly speaking, a computer policy

*Corresponding author. Center for Secure and Dependable Systems, University of Idaho, P.O. Box 441008, Moscow, Idaho 83844-1008, USA. Tel.: +1-208-885-4114, Fax: +1-208-885-6840, E-mail: jimaf@uidaho.edu.

should address security issues, i.e., CIA (Confidentiality, Integrity, Availability). While most publications do not discuss or provide a meaning of policy, a publication by Sterne [22] is devoted entirely to the concept of policy. Sterne indicated that a policy has several widely-used conflicting meanings and that a precise definition is needed to clarify security issues. Martin [19] pointed out that the concept of policy is difficult to define and discussed the network policy concept in greater depth.

In MILS (see Section 2), policies are rules that guarantee reliable message transmission between a system's entities. We use the concept *message* to refer to any data that has been encoded for transmission to or received from an entity (e.g., a method invocation, a response to a request, a program, passing a variable). The transmission mechanism can utilize shared memory, zero-copy message transport, kernel supported transport, TCP/IP, etc. An entity is the source or destination through which information can flow (e.g., user, subject, object, file, printer).

The security policies that MILS focuses on are those that impose constraints on the actions (operations) of entities in the system. This includes *access control* in addition to *information flow* restrictions. Access control policies ensure that entities access only those resources and services that they are entitled (authorized) to access, which prevents information damage by having restrictions on access to information; however, access control policies do not control how information is released after it has been read. Information flow policies [8,9] restrict the release of information after it has been read within a system or between systems. In other words, in addition to enforcing access control, information flow between entities must be monitored because a sequence of accepted operations can cause information flow between entities, thus resulting in an information leak. For the rest of the paper, we use the term *information flow* policies to refer to both types of policies.

The concept of information flow is fairly easy to understand, but enforcing information flow policies can be a difficult task. Information in MILS is controlled by mandatory information flow policies. A security policy refers to an information flow policy that describes which entities can pass certain messages to other entities. Security attributes associated with entities in the system are defined by the policy. Information flow decisions for entities are based on those attributes. This research focuses on one area of security: protecting confidential data from being revealed to unauthorized entities.

1.2. Why IEMP?

Security policies address different aspects of system security such as information flow, availability, auditing, and authentication. In this paper, we present the concept of an Inter-Enclave Multi-Policy (IEMP) for information access. We use the concept *security enclave* (coalition) to refer to a boundary for a group of entities that have the same security level. Entities in an enclave can communicate with one another according to an individual security policy that is responsible for that enclave. In a multi-enclave environment where several policies exist, entities in different security enclaves cannot interact with one another in a secure way without the existence of a mechanism that controls the interaction. In this paper, we propose an approach where all interactions between policies may be controlled by a global multi-policy that guarantees the integration of several heterogeneous systems. For example, in a coalition model, IEMP can integrate Army, Air Force, and Navy forces with a joint staff that ensures policy compliant interaction between the coalition members.

In many environments, an application or resource may be shared by several entities, with each entity having its own security constraints for the application. In such a diverse environment, a single consistent open framework for policy integration is needed to handle the conflicts and cooperation of policies. This is the role of an IEMP. MILS IEMPs are "policies about policies" that ensure the enforcement of end-to-end mandatory information flow security policies. The IEMP technique manages multiple security policies (i.e., controls the conflicts and cooperation of policies of different enclaves) that can be applied to a spectrum of high assurance systems. This allows the system security officer to manage the evolution of policies without modifying the applications to which these policies apply.

1.3. Multi-policy paradigm

In order to address the move toward the multi-policy paradigm that was first adopted by the United States Department of Defense (DoD) in 1993, we are applying the IEMP approach to the MILS system. The MILS system is a multi-policy security system that supports a variety of independent security enclaves [1,2]. A policy in the system can effectively deal with its enclave interactions (the entities that can communicate with one another in regards to that policy). When different policies in different enclaves communicate with one another, the complexity of guaranteeing no conflicts between policies greatly increases. Our goal is to enable the MILS system to effectively support secure information processing within multi-enclave-multi-policy environments.

The remainder of this paper is organized as follows: in Section 2, we describe the Multiple Independent Levels of Security and Safety (MILS) approach to high assurance system design for security- and safety-critical multi-enclave systems. We propose an Inter-Enclave Multi-Policy (IEMP) paradigm for information access in Section 3. Detecting and resolving policy conflicts is presented in Section 4. Section 5 discusses specifying individual policies and IEMPs. In Section 6, related research regarding security multi-policies is outlined. Finally, we conclude the paper and indicate future work in Section 7.

2. MILS architecture

High assurance systems are those that require convincing evidence that the system adequately addresses critical properties such as security and safety [13]. MILS is a joint research effort between academia, industry, and government led by the United States Air Force Research Laboratory with stakeholder input from the Air Force, Army, Navy, National Security Agency, Boeing, Lockheed Martin, Objective Interface Systems, Green Hills Software, Lynux Works, Wind River, General Dynamics, Raytheon, Rockwell Collins, MITRE, and the University of Idaho. The concept of the MILS architecture was created to simplify the process of the specification, design, and analysis of high assurance computing systems [24]. This approach is based on the concept of separation, as introduced by Rushby [20]. The concept of separation has been accepted in the avionics community and is a requirement of ARINC 653 [3] compliant systems. Through separation, we can develop a hierarchy of security services where each level uses the security services of a lower level to provide a new security functionality that can be used by higher levels. Limiting the scope and complexity of the security mechanisms provides us with manageable, and more importantly, evaluable implementations.

Within the MILS architecture, application layer entities are provided with the mechanisms to control, manage, and enforce their own application level security policies in a manner that ensures that the enforcement mechanisms are always invoked, non-bypassable, tamperproof, and evaluable. At this level we have both trusted application-level security services and untrusted applications. The MILS architecture assumes certifiable trust within the microprocessor, separation kernel, middleware services layer, and for the application-level security services. Thus, we assume a benign fault model within the microprocessor and real-time operating system (RTOS) but a malicious fault model for the untrusted applications. Benign faults will need to be addressed via traditional fault-tolerant diversity and redundancy, with fault containment procedures instituted within the microprocessor and RTOS. For example, illegal instructions and memory violations are assumed to be trapped and handled via the microprocessor and separation kernel, likewise for covert channels and residual data needing sanitization. Violations of information flow that cannot be detected and trapped in this fashion will need to be handled within the middleware services layer. Faults occurring within the middleware services layer (those that cannot be detected and trapped at lower levels) constitute an open issue currently being addressed by the MILS research community. At the application layer, execution is confined to the application partition, with limited communication to other partitions. All communication is monitored by the certified application layer security services and lower processing levels. Applications are assumed to be rogue and are confined to operating within their partition resources, with finite boundaries of space and time.

A MILS system isolates processes into partitions, which define a collection of data objects, code, and system resources. These individual partitions can be evaluated separately, if the MILS architecture is implemented correctly. This divide-and-conquer approach will exponentially reduce the proof effort for secure systems. A detailed description of the MILS approach is provided by Alves-Foss et al. [1,2].

3. Proposed model

The MILS system is a convenient environment for applying the IEMP approach for many reasons, including: different processes in the system enforce different security policies with different security goals in mind (e.g., confidentiality, integrity, and availability), the system deals with different users at different security levels, and MILS consists of separate components that interact with one another. Each component has its own functionality with its own security policy. To achieve security, our goal is to secure all interactions between the system's components using MILS *Security Policy Groups* (SPGs).

3.1. Policy architecture

Information access controls are the mechanisms involved in the mediation of every request to resources and data maintained by a system. Based on the security policy, they determine whether the request should be granted or denied. This mediation must be performed by a trusted component, the MILS *Policy Manager*. Figure 1 shows the policy architecture of MILS.

The *policy manager* makes access decisions in individual enclaves or between different enclaves, and the *policy database* stores the policies that the policy manager will need. The system security officer has the authority to specify security policies that are enforced by the system. Entities interact with the system to send requests through an entity interface. Auditing will be performed for entity requests, i.e., a request will be logged as a trace operation which will be used for analysis of activities in the system.

The policy manager is the policy enforcement mechanism that mediates message transmission between entities. Once an entity makes a request to pass information, the request will trigger the policies that are related to the requesting entity. The policy manager receives the request and identifies the policies that have been triggered. The policy manager is separate from the policy database, which makes the system flexible and simple; the system security officer will be able to change policies without modifying the enforcement mechanism.

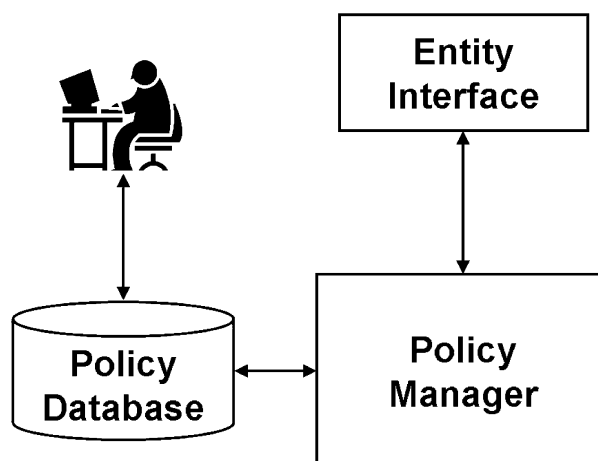


Fig. 1. Policy system architecture.

The policy manager is *consistent* and *complete*. It is consistent because an entity request is either accepted or denied but not both. This is due to the conflict resolution techniques that force the policy manager to make a decision. The policy manager is complete because for each entity request, there is a result (the access being accepted or denied).

Different policy models in the literature (e.g., Bell-LaPadula, Role-Based Access Control, Chinese Wall) have been developed to restrict information access. Although most systems are restricted to a single policy model to provide security [21], our proposed approach is capable of dealing with multiple policies from different models that are being enforced by the system. Different policies can all exist in one policy database. The policy manager checks the triggered policies and resolves potential conflicts (see Section 4). If the invoking entity is allowed to access another entity, then access is granted; otherwise it is denied. The policy manager is responsible for enforcing and monitoring the individual security policies and the multi-policies that are related to entities involved in the access.

3.2. MILS security policy groups

Kühnhauser [16] defined *policy groups* as a combination of a set of security policies and a set of policies that control inter-domain actions. An advantage of a policy group approach is that it composes a multi-policy system's security policies into a single structure and provides a single point of reference for the discussion of a system's security properties.

The approach taken in IEMP implements Kühnhauser's definition of different classes of actions in multi-policy systems but focuses on policy specification and enforcement methodologies. His definition is well suited for MILS, a multi-policy system that supports separate security policies for different enclaves in a diverse environment. An enclave sets a boundary for a group of entities that can communicate with one another according to an individual security policy responsible for that enclave. Each enclave has its own individual security policy that controls communication between entities that belong to the enclave. While an individual policy controls message communication within its enclave, inter-enclave multi-policies handle message communication between two or more enclaves. Enclaves can be arranged in a hierarchical tree structure and may exist across multiple processors (Fig. 2).

Any interaction between MILS entities is modeled as an entity e_1 accessing another entity e_2 through access operation op (*read*, *write*). $P(e_1, e_2, op)$ denotes the application of policy P to access (e_1, e_2, op) , so $P(e_1, e_2, op)$

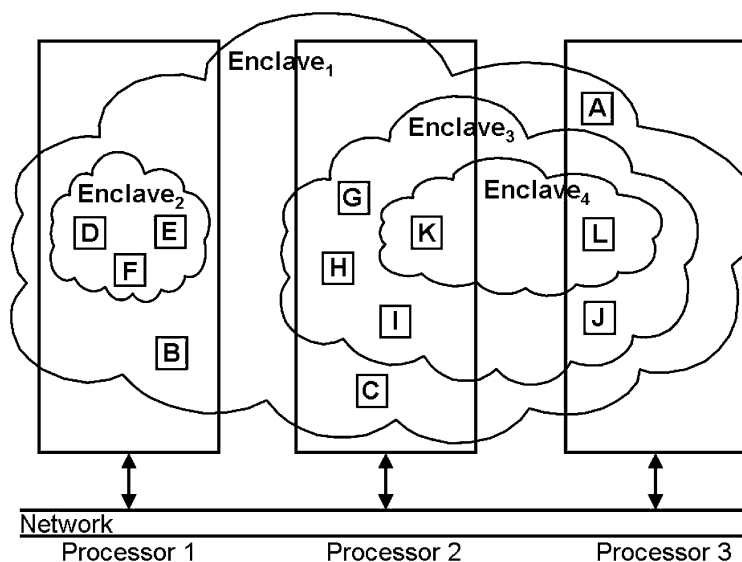


Fig. 2. MILS enclaves distributed over separate processors.

is of type *grant* or *reject*. Based on the sender's identity, recipient's identity, and some of the message content or type, a decision is made to *grant* or *reject* access. We assume that a policy should be able to respond to a certain request only to the entity that made the request (e.g., entity *A* should not receive information requested by another entity *B*). A policy should make a decision and respond to a request within a period of time specified by the system security officer. The system security officer assigns different time limits based on entity priority (importance).

3.3. Classes

$Enclave_P$ is used to denote the domain belonging to *P* which consists of all entities that are submitted to *P*. For any access (e_1, e_2, op) , a policy *P* will contain an access rule if and only if $e_1, e_2 \in Enclave_P$. $S_e = \{P | e \in Enclave_P\}$ denotes the set of security policies that have entity *e* within their enclave. $|C|$ is used to denote the cardinality of some set *C*, *I* a finite set of indices, and $\{P_i\}_{i \in I}$ a set of security policies.

In an enclave communication with a set of security policies $\{P_i\}_{i \in I}$, any access (e_1, e_2, op) in a multi-policy system with $e_1, e_2 \in \bigcup_{i \in I} Enclave_{P_i}$ belongs to one of the following three disjoint classes:

Class 1: $|S_{e_1}| = |S_{e_2}| = 1 \wedge S_{e_1} = S_{e_2}$

Class 1 identifies the case in which conflict-free interactions occur when both entity e_1 and entity e_2 belong to exactly one enclave (i.e., the same enclave) and are not members of any other enclave. Since no inter-enclave communication is required, a single policy *P* makes the access decision.

Class 2: $|S_{e_1} \cap S_{e_2}| = 0$

Class 2 identifies the case in which no security policy exists that has both entity e_1 and entity e_2 in its enclave; no security policy can provide the rule for interaction across multiple enclaves. An additional *completeness* policy is required to handle the communication. Two sub-classes exist:

(a) $|S_{e_1}| = 1 \wedge |S_{e_2}| = 1$

where each entity is a member of only one enclave.

(b) $\exists e \in \{e_1, e_2\} : |S_e| > 1$

where at least one of the entities is a member of more than one enclave.

Class 3: $|S_{e_1} \cap S_{e_2}| \geq 1 \wedge \exists e \in \{e_1, e_2\} : |S_e| > 1$

Class 3 identifies the case in which at least one policy provides an access rule for both entities and at least one of the involved entities is a member of more than one enclave. This may cause a conflict which requires a mediation policy to identify appropriate rules. *Inter-enclave multi-policies* are mechanisms that resolve such conflicts between two or more policies. Two different types of conflicts exist:

(a) $|S_{e_1} \cap S_{e_2}| = 1$

an *enclave conflict* where an entity is a member of more than one policy enclave.

(b) $|S_{e_1} \cap S_{e_2}| > 1$

a *rule conflict* where more than one policy exist for both entities that provide rules for the access.

A MILS *Security Policy Group* (SPG) is defined by combining the security policies, completeness policy, and conflict mediation into a single policy group. Let *I* be a finite index set and $\{P_i\}_{i \in I}$ be the set of regular security policies of a given multi-policy system. A security policy group $SPG = (\{P_i\}_{i \in I}, T, F, c)$ consists of the following: a set of security policies $\{P_i\}_{i \in I}$ implementing the security requirements for Class 1 accesses, a completeness policy *T* implementing the security requirements for Class 2 accesses, a conflict mediation policy *F* implementing the security requirements for Class 3 accesses, and a classification function *c* that for each access (e_1, e_2) , $e_1, e_2 \in \bigcup_{i \in I} Enclave_{P_i}$ produces the class (e_1, e_2) .

For any $e_1, e_2 \in \bigcup_{i \in I} Enclave_{P_i}$, *c* is defined as follows:

$$c(e_1, e_2) = \begin{cases} P_k & : |S_{e_1}| = |S_{e_2}| = 1 \wedge S_{e_1} = S_{e_2} = \{P_k\} \\ T & : |S_{e_1} \cap S_{e_2}| = 0 \\ F & : |S_{e_1} \cap S_{e_2}| \geq 1 \wedge \exists e \in \{e_1, e_2\} : |S_e| > 1. \end{cases}$$

The classification function is part of the policy manager that implements access mediation by overwriting the regular security policy call that is issued on every entity interaction. Any Class 1 interaction is directed to its regular individual security policy, Class 2 interactions are diverted to T , and Class 3 interactions are diverted to F .

3.4. Enclave examples

MILS consists of different components, each of which has its own security policy. Figure 3 shows an example of interactions between MILS components using MILS SPGs for four enclaves each with separate security policies. The outer enclave indicates an enclave of a global policy and the innermost one indicates a more restrictive policy. Assume that $Enclave_1$ represents the University, $Enclave_2$ represents the Research Office, $Enclave_3$ represents the College of Engineering, and $Enclave_4$ represents the Department of Computer Science. Assume an employee Karen, denoted K , who works full-time at the Department of Computer Science, is temporarily assigned to work at the Research Office. This temporary assignment is considered a process of crossing over policy boundaries. Once K logs in to the system at the Research Office, she becomes an entity within $Enclave_2$. When K tries to access a file, denoted W , within her original enclave, $Enclave_4$, it is considered inter-enclave access.

For now, let us ignore the policies of $Enclave_1$ and $Enclave_3$. Since $Policy_2$ does not have a rule for W in $Enclave_4$, and $Policy_4$ does not have a rule for K in $Enclave_2$, an IEMP SPG is needed to control the access.

- The SPG is $(\{Policy_2, Policy_4\}, T, F, c)$. S_K is $\{Policy_2\}$ and S_W is $\{Policy_4\}$.
- K 's access is classified by function c as Class 2 access: $|S_K \cap S_W| = |\{Policy_2\} \cap \{Policy_4\}| = |\emptyset| = 0$.

Therefore, policy T is selected for the access. The exact role of policy T will reflect the method of inter-enclave access in MILS. For example, T could map K in $Enclave_2$ to K in $Enclave_4$ giving K the ability to have access rights in both $Enclave_4$ and $Enclave_2$.

In the case of policy conflicts, consider the previous example, but this time take all the policies of the enclaves into account.

- Entities in $Enclave_2$ are indirect members of $Enclave_1$ and entities in $Enclave_4$ are indirect members of $Enclave_3$ and of $Enclave_1$.
- The SPG is $(\{Policy_1, Policy_2, Policy_3, Policy_4\}, T, F, c)$.

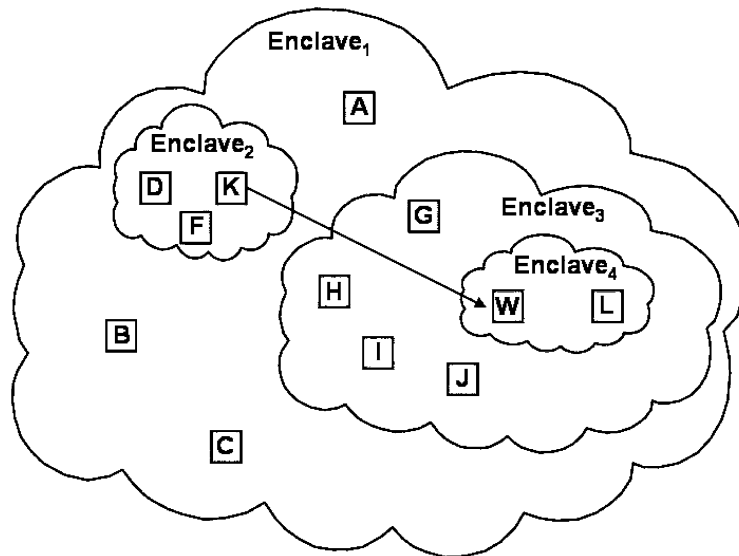


Fig. 3. A structure of policy enclaves.

- When K is in $Enclave_4$, S_K is $\{Policy_1, Policy_3, Policy_4\}$.
- When K is in $Enclave_2$, S_K is $\{Policy_1, Policy_2\}$.
- S_W is $\{Policy_1, Policy_3, Policy_4\}$.
- K 's access is classified by function c as Class 3 access: $|S_K \cap S_W| \geq 1$ in both cases, and each S contains more than one policy.

Therefore, policy F is selected for the access. The exact role of policy F will reflect the method of resolving conflicts in inter-enclave access in MILS. There are two situations which can give rise to these potential conflicts:

- 1. Rule conflicts:** These occur in Class 3 from Section 3.3 ($|S_K \cap S_W| > 1$). When K accesses her file while in $Enclave_4$, a rule conflict would arise between policies $\{Policy_1, Policy_3, Policy_4\}$. One choice F could define is that the innermost policy overrides the outermost one; and therefore the innermost policy (i.e., $Policy_4$) will control K 's request and make the access decision. With hierarchical policies, it is often easy to make such a determination.
- 2. Enclave conflicts:** These occur in Class 3 from Section 3.3 ($|S_K \cap S_W| = 1$). When K accesses her file while in $Enclave_2$, an enclave conflict would arise between $Enclave_2$ and $Enclave_4$. Since no inner policy is able to make the decision, the global policy $Policy_1$ will control K 's request and make the access decision. Or, as provided by policy T , K in $Enclave_2$ could be mapped to K in $Enclave_4$. When the conflicts are not hierarchical, *a priori* decisions must be made for any potential enclave conflict.

4. Policy conflicts

The IEMP technique allows actions to be specified to resolve policy conflicts. With the coexistence of different policies in a diverse environment, interactions between the system's entities can result in policy conflicts. One policy may allow certain operations to be performed by an entity while another policy may not. IEMPs should not only control the cooperation of enclave individual policies to detect conflicts, but also take appropriate actions to resolve these conflicts.

As mentioned in Section 3.2, Class 3 identifies the case in which at least one policy provides a rule for the access and at least one of the involved entities is a member of more than one enclave. IEMPs identify two types of conflicts: an *enclave conflict* where an entity is a member of more than one policy enclave and a *rule conflict* where more than one policy exist that provide rules for the access.

Potential conflicts could be identified at the time policies are being defined, but this tends to be time-consuming and therefore inefficient. A better approach is to have rules that control the interaction once a conflict has been detected. Many conflicts will be resolved if each entity is assigned an explicit priority by the system security officer based on its position in the hierarchy (i.e., importance). Once a conflict has been detected, an IEMP refers to the priority of the involved entities, and the policy of the entity that has the highest priority will be considered. Another approach is to give priority to the innermost policy over the outermost one. The innermost policy will become stronger and resolve the conflict. For example, when a conflict arises regarding a CS student in the CS Department, the policy of the CS Department will refine (add to) those of the University.

An IEMP should immediately decide how to resolve policy conflicts in a proper way. Resolving conflicts involves determining which policy will take precedence or what actions will resolve the conflicts. Each enclave policy is assigned a priority and a creation/modification date. When inter-enclave policies have conflicting rules, the policy that has the highest priority takes over. If two policies have the same precedence, then the policy that was created/modified on a later date will take precedence over the policy that was created/modified on an earlier date. It must be noted that IEMPs are not allowed to rewrite enclave policies to resolve the conflict; this could lead to more conflicts. Instead, the system security officer can modify or remove existing policies and add new policies.

When policies conflict with one another, an IEMP should decide on a resolution. To perform policy conflict analysis, static and/or dynamic analysis could be performed. Static analysis occurs when conflicts are detected at compile-time; dynamic analysis occurs at run-time. Our policy model in MILS performs static analysis. Although most policy models perform static analysis, some recent research [4,11] addresses dynamic policy analysis.

5. Policy specification

Although various languages in the literature have been proposed to specify policies for different purposes, a standard language does not yet exist for the policy community to use. The MILS architecture provides a language for describing security policies. Such a policy language is flexible enough to allow customizing of security policies to reflect user needs. Security policies are managed by the system security officer who defines and specifies security policies to apply to the system and who also modifies these policies when the system configuration changes.

We declare policies using MPL (*MILS Policy Language*). MPL, still in the developing process, consists of natural language English-like notations that are functional enough to allow policies to be specified as constraints on allowable actions in the environment. Figure 4 shows an IEMP example in MPL, based on Bell-LaPadula [6]. Although policies represent how an entity should behave, whether the entity complies with the policy depends on the enforcement mechanism, i.e., the policy manager. In order to limit the applicability of a valid policy, we suggest having certain constraints on entities based on information such as date, time, access history, and/or location. For example, one could use date and time to restrict the use of the system between the days of Monday and Friday and the hours of 8:00 am and 5:00 pm.

The policies specified in MPL are translated (refined) into Prolog, i.e., they are transformed from high-level abstract specifications to low-level operational policies that MILS can enforce. The policy manager uses a knowledge base and reasoning techniques for security. It reads policies and stores them in a Prolog knowledge base. Prolog policies will effectively enforce security because Prolog tends to be an appropriate language for expressing policies due to its simple declarative nature. Figure 5 shows an IEMP example in Prolog that is a subset of what will be implemented. The policy is based on Bell-LaPadula [6].

6. Related work

Although all the following meta-policy concepts express that meta-policies are policies about policies, each concept was introduced to target a specific area of interest. The meta-policy concept “policies about policies” was introduced by Hosmer [14,15]. Hosmer showed that the conflict resolution process can be simple no matter how many different policies are included. Kühnhauser [16] introduced meta-policies cooperating between policies from different domains. Kühnhauser followed Hosmer’s views of meta-policies, but the focus of his work was on policy coordination. Kühnhauser and von Kopp Ostrowski [17] engaged meta-policies to construct a formal framework that supports multiple policies. Their implementation used “custodians” whose functionality is similar to that of the policy manager used in our work. Bell [5] applied meta-policies to policy negotiation with the design of a formal “multipolicy machine”. Bell considered the composition of two policies as a function, called *policy combiner*, and introduced *policy attenuation* to allow the composition of conflicting security policies. Lupu and Sloman [18] used meta-policies as a way to specify logical predicates over different policies. Meta-policies were introduced because conflicts may arise between policies applied to the same object processing different management functionalities, or policies are applied to objects which are members of several domains. More recently, Belokosztolszki and Moody [7] presented meta-policies to check policies at specification time. In addition, Granville et al. [12] proposed PoP (Policy of Policies), which defines how policies might be reorganized or disabled in response to the detection of run-time conflicts. The concept of a *security enclave* that we defined in this paper (see Section 1.2) is similar to that defined by DISA (Defense Information Systems Agency) [10], Zellmer [25], and Watro and Shirey [23].

7. Conclusions and future work

This paper outlines an inter-enclave multi-policy paradigm in MILS. MILS consists of different components, each of which has its own security policy. The existence of an inter-enclave multi-policy framework across several components becomes essential for cooperation and conflict resolution between individual policies. It is clear that

```

iemp enclave1-enclave2
{
  security-levels:
    level = {U, C, S, TS}; //Unclassified, Confidential, Secret, Top Secret
  classifications: //ordering of classifications
    level = {U≤C, U≤S, U≤TS, U=U, C≤S, C≤TS, C=C, S≤TS, S=S};
  entity-operations:
    //BLP simple security property: read down
    //cl: classification hierarchy modeling a lattice
    //am: access matrix
    read-access(ei-enclave1,ej-enclave2,op) :
      less-than-or-equal(cl(ej-enclave2),cl(ei-enclave1)), equal(op,read),
      in(op,am(ei-enclave1,ej-enclave2)), equal(access(ei-enclave1,ej-enclave2),false);
    //BLP *-property: write up
    write-access(ei-enclave1,ej-enclave2,op) :
      less-than-or-equal(cl(ei-enclave1),cl(ej-enclave2)), equal(op,write),
      in(op,am(ei-enclave1,ej-enclave2)), equal(access(ei-enclave1,ej-enclave2),true);
  policy-restrictions: //enforce enclave restrictions (e.g., monday≤sys.day≤friday)
    if (enclave1-restrictions || enclave2-restrictions) then restrictions = true;
  entity-state: //cl: classification hierarchy modeling a lattice, //am: access matrix
    entities={e1-enclave1,e2-enclave1,e3-enclave2,e4-enclave2}; //controlled by IEMP
    cl(e1-enclave1)=TS, cl(e2-enclave1)=TS, cl(e3-enclave2)=S, cl(e4-enclave2)=S;
    am(e1-enclave1,e2-enclave1)=am(e2-enclave1,e1-enclave1)=
      am(e1-enclave1,e1-enclave1)=am(e2-enclave1,e2-enclave1)={read,write}, //enclave1
    am(e3-enclave2,e4-enclave2)=am(e4-enclave2,e3-enclave2)=
      am(e3-enclave2,e3-enclave2)=am(e4-enclave2,e4-enclave2)={read,write}, //enclave2
    am(e1-enclave1,e3-enclave2)=am(e1-enclave1,e4-enclave2)=
      am(e2-enclave1,e3-enclave2)=am(e2-enclave1,e4-enclave2)={read},
    am(e3-enclave2,e1-enclave1)=am(e3-enclave2,e2-enclave1)=
      am(e4-enclave2,e1-enclave1)=am(e4-enclave2,e2-enclave1)={write}; //iemp
    if cond is satisfied then (access(ei-enclave1,ej-enclave2)=true, flag1=true)
    else (access(ei-enclave1,ej-enclave2)=false, flag2=true);
    if (flag1 && flag2) then conflict-priority;
  conflict-priority:
    if (enclave1-priority > enclave2-priority) then resolve = enclave1-priority
    else if (enclave1-priority < enclave2-priority) then resolve = enclave2-priority
    else creation-date;
  creation-date:
    if (enclave1-date > enclave2-date) then resolve = enclave1-date
    else if (enclave1-date < enclave2-date) then resolve = enclave2-date
    else resolve = enclave1-date;
}
cond: The simple and *-properties of Bell-LaPadula. The operation of a read defining ≤r implies “cl(o) ≤1 cl(s)” and
the operation of a write defining ≤w implies “cl(s) ≤1 cl(o)”.

```

Fig. 4. An IEMP example in MPL.

```

% Entities
entity(jack). entity(penny). entity(camille). entity(mike). entity(trudy). entity(adrian).

% Enclaves
enclave(1). enclave(2). enclave(3). enclave(4).

% Security Clearance: 1 = Unclassified, 2 = Confidential, 3 = Secret, 4 = Top Secret
% classification(entity,enclave,securityclearance).
classification(jack,1,4). classification(penny,1,4). classification(camille,1,4).
classification(mike,2,3). classification(jack,2,3). classification(trudy,3,2).
classification(adrian,4,1).

% One enclave, read operation
allow(Entity1,Enclave1,read,Entity2,Enclave2) :-
    Enclave1 = Enclave2,
    entity(Entity1), entity(Entity2), enclave(Enclave1), enclave(Enclave2).

% One enclave, write operation
allow(Entity1,Enclave1,write,Entity2,Enclave2) :-
    Enclave1 = Enclave2,
    entity(Entity1), entity(Entity2), enclave(Enclave1), enclave(Enclave2).

% Different enclaves, read operation
allow(Entity1,Enclave1,read,Entity2,Enclave2) :-
    entity(Entity1), entity(Entity2), enclave(Enclave1), enclave(Enclave2),
    dominate(Entity1,Enclave1,Entity2,Enclave2).

% Different enclaves, write operation
allow(Entity1,Enclave1,write,Entity2,Enclave2) :-
    entity(Entity1), entity(Entity2), enclave(Enclave1), enclave(Enclave2),
    dominate(Entity2,Enclave2,Entity1,Enclave1).

% Dominate relation
dominate(X,C,Y,V) :-
    classification(X,C,CL1), classification(Y,V,CL2), CL1 ≥ CL2.

% Sample Run
| ?- allow(jack,1,read,adrian,4).
true ?
yes
| ?- allow(adrian,4,write,jack,1).
true ?
yes

```

Fig. 5. An IEMP example in Prolog.

in a multi-enclave environment, multiple security policies need to interact. We propose an inter-enclave multi-policy technique that manages multiple security policies within heterogeneous systems and define IEMPs as sets of actions to control the interactions between different policy enclaves. IEMPs are significant in high assurance multi-enclave systems because they control the processing and coordination of policies.

We describe an enclave policy specification in MPL that is translated into Prolog. We believe that implementing information control using MPL and Prolog IEMPs is a good approach. However, it is important to acknowledge

that establishing more policy constraints in more complex scenarios is necessary to provide more flexibility in supporting security policies. Further research will include non-constraint (dynamic) policies that indicate what should happen in certain situations (e.g., intrusion detection) as opposed to the current constraint (static) policies that restrict actions of an entity (*accept, deny*). Other issues remain to be investigated, including developing a graphical tool to specify individual policies and visualize interactions between inter-enclave policies.

Acknowledgements

We wish to acknowledge the US Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) for their support. This material is based on research sponsored by AFRL and DARPA under agreement number F30602-02-1-0178. The US Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL, DARPA, or the US Government. We also wish to acknowledge the anonymous reviewers and journal editors for reviewing this paper.

References

- [1] J. Alves-Foss, W.S. Harrison, P. Oman and C. Taylor, The MILS architecture for high assurance embedded systems, *International Journal of Embedded Systems* 2(1) (2006).
- [2] J. Alves-Foss, C. Taylor and P. Oman, A multi-layered approach to security in high assurance systems, in: *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 2004.
- [3] Avionic application software standard interface (Draft 3 of Supplement 1) (Specification ARINC 653), 2003. ARINC Standards.
- [4] A. Bandara, E. Lupu and A. Russo, Using event calculus to formalise policy specification and analysis, in: *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, 2003, pp. 26–39.
- [5] D. Bell, Modeling the “multipolicy machine”, in: *Proceedings of the New Security Paradigms Workshop*, 1994, pp. 2–9.
- [6] D. Bell and L. LaPadula, Secure computer systems: Unified exposition and MULTICS interpretation, Technical Report ESD-TR-75-306, MITRE Corporation MTR-2997 Rev. 1, March 1976.
- [7] A. Belokosztolszki and K. Moody, Meta-policies for distributed role-based access control systems, in: *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002, pp. 106–115.
- [8] M. Bishop, *Introduction to Computer Security*, Addison Wesley Professional, 2004.
- [9] D. Denning, A lattice model of secure information flow, *Communications of the ACM* 19(5) (1976), 236–243.
- [10] Defense Information Systems Agency (DISA) enclave security technical implementation guide (Version 3, Release 1), July 2005.
- [11] N. Dunlop, J. Indulska and K. Raymond, Dynamic conflict detection in policy-based management systems, in: *Proceedings of the 6th International Enterprise Distributed Object Computing Conference*, 2002, pp. 15–26.
- [12] L. Granville, A. Coelho, M. Almeida and L. Tarouco, PoP – An automated policy replacement architecture for PBNM, in: *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002, pp. 140–145.
- [13] M. Heimdahl and C. Heitmeyer, Formal methods for developing high assurance computer systems: Working group report, in: *Proceedings of the 2nd IEEE Workshop on Industrial-Strength Formal Specification Techniques*, 1998, pp. 60–64.
- [14] H. Hosmer, Metapolicies I, *ACM SIGSAC Review – Special Workshop on Data Management Security and Privacy Standards* 10(2–3) (1992), 18–43.
- [15] H. Hosmer, The multipolicy paradigm for trusted systems, in: *Proceedings of the New Security Paradigms Workshop*, 1993, pp. 19–32.
- [16] W. Kühnhauser, Policy groups, *Computers & Security Journal* 18(4) (1999), 351–363.
- [17] W. Kühnhauser and M. von Kopp Ostrowski, A framework to support multiple security policies, in: *Proceedings of the 7th Annual Canadian Computer Security Symposium*, 1995.
- [18] E. Lupu and M. Sloman, Conflicts in policy-based distributed systems management, *IEEE Transactions on Software Engineering* 25(6) (1999), 852–869.
- [19] J. Martin, Policy-based networks, Part no. 806-3718-10, Revision 01. Sun Microsystems Inc., 901 San Antonio Road, Palo Alto, CA 94303, USA, October 1999.

- [20] J. Rushby, Design and verification of secure systems, in: *Proceedings of the 8th ACM Symposium on Operating System Principles*, 1981, pp. 12–21.
- [21] R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. Andersen and J. Lepreau, The Flask security architecture: System support for diverse security policies, in: *Proceedings of the 8th USENIX Security Symposium*, 1999, pp. 123–139.
- [22] D. Sterne, On the buzzword “security policy”, in: *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 1991, pp. 219–230.
- [23] R. Watro and R. Shirey, Mapping mission-level availability requirements to system architectures and policy abstractions, in: *Proceedings of the DARPA Information Survivability Conference & Exposition II*, Volume 1, 2001, pp. 189–199.
- [24] P. White, W. Vanfleet and C. Dailey, High assurance architecture via separation kernel, October 2000, draft.
- [25] D. Zellmer, Multi-level security: Reality or myth, March 2003, GSEC practical requirements v.1.4.b.