ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Engineering Applications of Artificial Intelligence ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Engineering Applications of ARTIFICIAL INTELLIGENCE

www.elsevier.com/locate/engappai

# Dynamic security reconfiguration for the semantic web

Juan Jim Tan, Stefan Poslad*

*Department of Electronic Engineering, Queen Mary, University of London, Mile End Road, London E1 4NS, UK*

Accepted 12 August 2004

**Abstract**

There is a plethora of security standards for protecting network services, specified by numerous standards consortia. These standards support different security requirements and use various syntaxes to represent the security information for different software infrastructures and applications. Open systems often require a more sophisticated security analysis and configuration to safeguard distributed services. A security framework constituting both semantic and meta-reasoning models is investigated in order to reason about the security requirements and security operation of interacting entities within open service environments. Security requirements are defined using security profiles that describe the interlinking of security policies to instances of services. Meta-reasoning refers to the reflection at a conceptual level at which the domain knowledge (ontology) is separated from the control knowledge (profiles): systems can manage and reconfigure themselves without affecting their underlying implementation. Such reasoning is particularly useful within open service infrastructures as it enables us to detect, analyse and resolve multiple-policy conflicts, to decide if a change in the environment necessitates a security reconfiguration, and to decide if a suitable level of security interoperability between heterogeneous systems is achievable. This paper describes a meta-reasoning model for semantic open service environments, an application and an evaluation of the framework and its performance.


*Keywords:* Reconfigurable security; Policy; Semantic management; Interoperability

## 1. Introduction

E-commerce services are becoming more heterogeneous and dynamic. In order to offer such services for use within e-business environments, security plays a mandatory part. The security issues arising from open service, multi-domain interaction are very complex to manage and support. This paper focuses on the policy layer part of this system in which dynamic security profiles and policies can be analysed to provide security decision support and reconfiguration information to the application layer. A dynamic security reconfiguration model has been developed that uses policy-based security profiles for making logical and knowledge-based decisions within open service environments and it uses a layered holistic model for describing security for open environments (Tan et al., 2004a). The reasoning model is useful in policy-based systems as it provides a dynamic model whereby rules and their execution can be related to changing assertions and it can be applied to heterogeneous services to support the resolution of policy conflicts.

### 1.1. Motivation for supporting dynamic security reconfiguration

In critical business applications, security plays a vital role in providing users and businesses with the confidence to interact safely. As services become more pervasive, ubiquitous, open and highly available, service processes that define the operation sequences or workflow of these systems will also need to be highly configurable. Having different application domains

*Corresponding author. Tel.: +44 20 7882 3754; fax: +44 20 7882 7997.

*E-mail addresses:* stefan.poslad@elec.qmul.ac.uk, juanjim.tan@elec.qmul.ac.uk (J.J. Tan).

0952-1976/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2004.08.036

within open environments causes a potential exponential increase in the number of heterogeneous security models that may need to interoperate with one another. In achieving security for such business environments, configurable systems using policy-driven approaches seem a useful approach. There are raft of research and development issues in this area:

- weighing up common and domain knowledge from different sources in order to improve the decision making;
- defining an independent *dynamic reconfiguration model* for discerning semantic-based security profiles that is modular, easy to manage and useable with different applications;
- analysing the strengths and weaknesses of different reasoning models to manage policies;
- separation of application and control modes to allow applications to be dynamically managed through a purely declarative manner such as policies at the control level—developers can thus focus on the representation of policies and business process logic rather than on performance issues;
- security requirements and policies may not be explicitly defined making it difficult to agree on an agreed configuration;
- different stakeholders' viewpoints of security can vary depending on different applications;
- a profusion of in-house incompatible configurations exist that can prevent end-to-end interoperability.

The general motivation of this paper is to provide a security model that can support the management, reconfiguration and reasoning about semantic-based security profiles, in which heterogeneous services can dynamically discover security information to enable different systems to communicate securely.

## 2. Related work

### 2.1. Perspectives on policy architectures

The management of large multi-domain distributed systems must reflect the distribution of the systems being managed. Management domains provide the means for partitioning responsibility by *grouping* objects in order to specify policies. *Access control policies* specify the operations a group of *subject* objects is permitted to perform on a group of *target* objects. Most research has focused on policy specification languages such as Ponder (Damianou et al., 2001), TPL (Herzberg et al., 2000), and PDL (Lobo et al., 1999) that define rules for policy management. Some of these policy-based management technologies have developed algorithms to handle conflict and resolution, such as the KAoS service (Bradshaw et al., 1997), and others such as Chomicki and Lobo (2001) and Marshall and Mckee (2001). As the scope of these policies is specified in terms of management domains, access control decisions are mainly based on the authenticated domain membership of the subject and can be difficult to apply to heterogeneous, dynamic, open service environments.

Other applications of policy-driven approaches such as Nomura et al. (1999) are policy-based management architectures that address a reduction in the total cost of ownership by providing automation to manage enterprise network device configurations. A policy-based management framework that supports automated policy deployment and flexible event triggers permitting dynamic policy configuration is presented in Lymberopoulos et al. (2002).

The Secure and Open Mobile Agent (SOMA) system describes the use of policy management using agent systems (Corradi et al., 2001). SOMA supports agent interoperability by means of the CORBA and MASIF standards and the application of Ponder (Damianou et al., 2001), but it is applied only in homogeneous (SOMA) environments and has not focused on multi-agent multi-domain (MAMD) environments. In the SOMA architecture, users may specify policies that are stored in a policy repository such as LDAP. A coordinator can then periodically check the repository for policy changes and update the relevant target environments. Subsequently, authorisation and obligation enforcement services are used to establish the adherence of the system behaviour based upon the specified policies. The resources of the SOMA environment are encapsulated and static. This is in contrast to the way in which resources are often expressed in MAMD systems—here they are fundamentally represented as autonomous agents or at least agent wrappers that are openly accessible, interchangeable, interoperable, and independent of lower-level services such as message transport services.

### 2.2. Security ontologies

The concept of an ontology in its broad sense covers a range of models ranging from flat sequential syntactical models as advocated by the IETF, W3C byte stream protocols, hierarchical syntactical models (XML) and dictionaries of security terms, to more expressive logic-based frameworks based on RDF and DAML. Each of these is considered in turn.

Firstly, *taxonomy dictionaries* (TD) describe security concepts as terms that can be analysed by a particular application. TDs are often used by intrusion detection and trust-based systems. In the area of standardisation, trust first became an issue almost 20 years ago (Department of Defence (DoD), 1985). Other specialised trust management solutions appeared such as W3C

PICS (Miller et al.) that was used to define formats and to distribute meta-data labels for the description of web documents. AT and T have developed policy maker (Blaze et al., 1996) that binds access rights to an owner of public key using certificates. IBM recognizes that trust is at the centre of e-business so it has developed a Java-based trust establishment module and a complementary control language (Herzberg et al., 2000). An extensive survey on trust has been published by Grandison and Sloman (2000). This survey defines trust informally. The main advantage of taxonomy dictionaries is that security concepts are explicitly defined. There exists a direct mapping between a particular action and an entry in a dictionary. Conversely, the major drawback of this approach is the difficulty of representing every single concept and action.

Secondly, in *XML-based approaches* the semantics of an XML document is explicitly encoded within the document using tags as identifiers and as such, it has the potential to support a finely grained security architecture. Until recently, most of the security systems created around XML standards have focused on protecting the transmission of documents. For example, simple object access protocol SOAP uses XML to encode messages to send across a network. As such, XML messages can be protected using HTTPS and SSL—these support confidentiality, integrity and authentication. Recent XML-based security standards include SAML, XMLSig and XKMS.

Thirdly, *policy-based and access control solutions* support the management of large multi-domain distributed systems. Management of domains and entities are partitioned into groups based upon membership details. In distributed security, trust must be decentralised to support verification from multiple-domain servers such as PAS in the SESAME architecture (Kaijser et al., 1994). Access control policies are specified in terms of domain membership rather than individual identities. Hence, the performance of the verification of domain membership can be critical for open systems. Similarly, other related work employing security agents on a per-node basis (Yialelis and Sloman, 1996) has been used to support secure communication, rights delegation and authentication.

Finally, various *security ontologies* that can represent security information in an intelligible manner have been examined. Using semantic models such as RDFS, entities can interpret security information more correctly.

Public key infrastructure or PKI is currently one of the most widely available authentication and certification frameworks used by industry. However, there exist numerous vendors specific PKIs such as the SET protocol PKI, Verisign PKI, etc. They are often not compatible because they use different types of certificates, use signatures in different ways and they manage keys differently. Users and suppliers may also need to hold multiple certificates because related certificates themselves vary by, certificate type, cipher type, by version within a type and by the use of extension fields within a type.

*Semantic service adaptable* specifications refer to XML-based representations that are easily convertible into richer semantic notations such as RDF/RDF-S. In Table 1, non-XML-based technologies can limit the promotion of a shared semantic security model amongst heterogeneous entities. The analysability and expressivity of such representations are further hampered without an ontological approach. When analysing data structures, an ontology can simplify the reasoning and validation of dependencies between concepts. In addition, it can promote the representation of behaviours of complex environments, the definition of multi-level abstractions and the extensibility to add new concepts.

### 2.3. Meta-reasoning models

There are a number of trade-offs using conventional reasoning mechanisms. Tools based on Expert Systems usually focus on a type of application and do not provide suitable knowledge representations for use across different domains. High-level programming languages provide decision making through runtime execution but this is static and can be difficult to manage.

The use of expressive formalisms may provide the necessary aspects for logical deductions to be made. Logic-based approaches have many advantages, they can provide representation formalisms and semantics drawn from a knowledge base. Furthermore, logics can provide the soundness, completeness and provability of properties to enable automated concrete decision making. However, in practice much logic-based reasoning relies on truth values and does not support semi-decidability (Harmelen, 1989). Nevertheless, other logics have been developed offering more expressivity and inferential strength such as modal and epistemic logics.

In this paper, meta-level architectures for managing and discerning policy-based service security are examined. As the name suggest, meta-level systems can be viewed as an architecture that separates domain knowledge (what the system knows) and control knowledge (how the system uses this knowledge) in Harmelen (1989). There are a number of advantages of this type of architectures: it promotes a loosely coupled approach to designing and managing knowledge outside the implementation system, and permits the sharing of knowledge such as ontologies with different systems. Meta-level architectures are classified into object, mixed and meta-levels. The intrinsic characteristic of this sort of system allows reasoning to take place at the application (object) level, at an abstract (meta) level where it controls the

Table 1
Summary of existing solutions

| Specification | Functionalities | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Confidentiality | Authentication | Integrity | Authorisation | Non-repudiation | Credential management | Interaction protocols | Policies | Trust delegation | Semantic service adaptable |
| XML encryption | ✓ | | ✓ | | | | | | | |
| XML signature | | ✓ | ✓ | | ✓ | | | | | |
| XACML | | ✓ | ✓ | ✓ | ✓ | | | ✓ | | ✓ |
| SAML | | ✓ | ✓ | ✓ | | | ✓ | | | ✓ |
| XKMS | | | ✓ | | | ✓ | ✓ | | | |
| SESAME | ✓ | ✓ | ✓ | ✓ | | | | ✓ | | ✓ |
| PICS | | | ✓ | ✓ | ✓ | | | ✓ | | |
| P3P | | | | ✓ | | | | ✓ | | ✓ |
| Keynote | | | ✓ | ✓ | | ✓ | | ✓ | ✓ | |
| S/MME/PGP | ✓ | ✓ | ✓ | | | | | | | |

behaviour of the application, or mixed levels where it performs computation at both the object and meta - levels.

At an object level, the reasoning system refers to an application with a fixed computational cycle, and a reasoner that generates the decisions to be carried out at various fixed points within this cycle. The operational aspect of these systems is static and is only capable of reacting or interacting within its local domain, such as a dedicated point of sale or non-autonomic based systems. On the contrary, a meta-level reasoning system decides the behavioural aspect of the system dynamically, where it is able to learn and react to new actions and goals. Meta-level reasoning features extensible domain knowledge (ontology) and control knowledge such as profiles (processes and policies), which are distinctively separated. Some examples include agent-based systems for service composition or supply chain management in which agents interacts with diverse services and can dynamically adapt to changing requirements. For these reasons, the scenario given in Fig. 4 clearly explains the need for a Conference Organising Agent (COA) to dynamically adapt and change its process logic to meet differing requirements when interacting with more than one domain of entities such as hotels, restaurants, venues and banks. Without recourse to meta-level or mixed level reasoning architectures, the COA may be incapable of accomplishing its goal if a pure object level reasoning system is adopted.

In this work a mixed level mechanism is used. The domain knowledge refers to the common security upper ontology that can be shared with a multitude of services and users. The control knowledge represents the security profiles. The model is categorised at a mixed level because all interpretation does not purely operate at the meta-level. The environment, preferences and operational permissions and request still rely on the object level. As a result, service-based reasoning models can be generally viewed to be part of a mixed level, where certain problems are still solved at the application level.

### 2.4. Autonomic computing

Autonomic computing has interestingly incited many challenges to the development of self-managing systems that can be dynamically reconfigured to meet demanding needs. Within the realm of security and open environments, systems that need to interoperate securely can benefit from autonomic computing. Horn of IBM has presented the theme and importance of autonomic computing and its impact in the next era of computing such as the semantic web; pervasive computing and ubiquitous systems. Systems are capable of reconfiguring operational functions without affecting their

underlying implementations. A system does not need to be pulled offline to make changes, but can be assimilated with new goals through policy alteration. Along with ontologies, semantics are able to provide what syntactic-based systems cannot deliver; extensibility and analysability, such as modelling functions at different levels of abstraction, easing the addition of new concepts and cross-referencing of conceptual terms (Bradshaw et al., 1997). Autonomic computing or self-managing systems have four levels: self-configuring, self-healing, self-protecting, and self-optimising. It is not to a great extent known as a technology, but as characteristics whereby diverse technologies can benefit from its merits. Steps in achieving autonomic systems have been outlined in Ganek and Corbi (2003). Here, five levels from manual to autonomic have been proposed for evolving a non-autonomic infrastructure towards an autonomic one. In Bantz et al. (2003) and Haas et al. (2003), personal computing and service deployment are set in the context of autonomic computing, where end users and e-businesses can gain from leveraging self-managing methods. Autonomic computing is complementarily related to technologies in the field of semantic web such as Agentcities (global agent testbed) and globus grid (Foster et al., 2001) where heterogeneous systems are able to interoperate, mediate and self-manage from the level of consumers to businesses. Autonomous agents provide the functionality for independently interacting with and managing resources based upon the user requirements and for mediating atomic services to form virtual enterprises. Security and autonomic computing are important features to realise many real-time e-business implementations, where self-managing methods are able to detect, react and mobilise resources for addressing threats (Chess et al., 2003).

## 3. Dynamic security reconfiguration model

The development of a holistic ontology is not an end itself, it provides the means by which security services and software such as agents can advertise and exchange security-related information and it acts as a model for the management of security processes and services. An ontological model promotes the interoperability between disparate security systems, providing a framework to support secure transactions across heterogeneous multi-domain boundaries. In developing an ontological model, an abstract model that was not directly dependent on particular security mechanisms or specifications is specified—making the model more maintainable. The ontological model (Tan et al., 2004a), Fig. 1, consists of the following:

- *conceptual layer*: defines the properties and relations between security, trust and privacy related concepts;
- *reification layer*: comprises the following sub-layers:
  - *service description layer*: the means for security processes to be hooked into service processes.
  - *policy layer*: provides the means for defining security rules and constraints;
  - *trust layer*: provides the means for defining trust implementations within systems to enable soft security interoperability between disparate applications. This is however not the focus of this paper and is not discussed further.

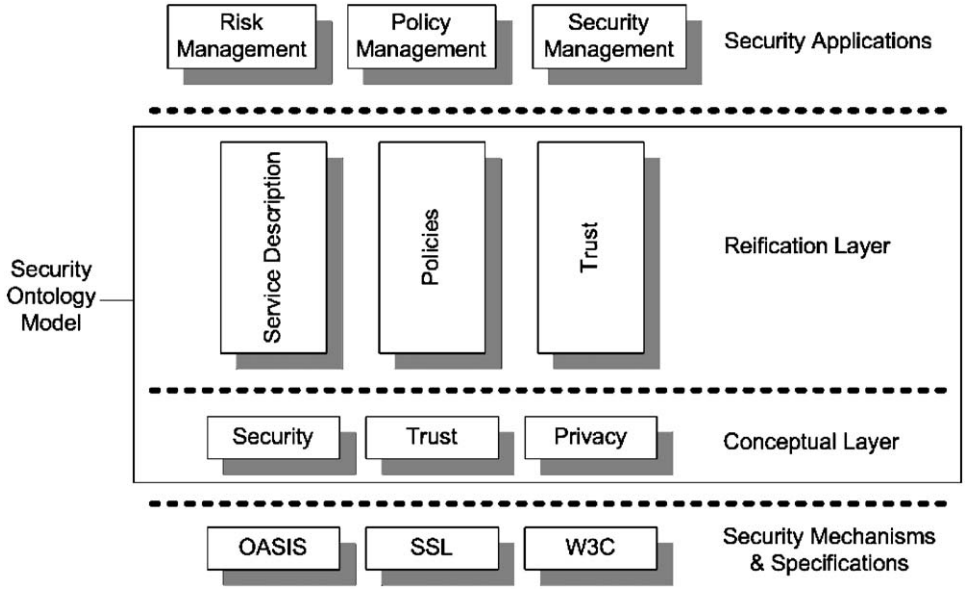


Fig. 1. An ontological model to support security interoperability.

These are separated from:

- *security mechanisms*: specific instances of security concepts, policies, and service entities as defined in existing security standards;
- *security applications*: that make commitments to use the security ontology within specific application domains.

This separation enables the security model to be independent of the application requirements and the use of specific security mechanisms. In the following section this model is defined in more detail.

An abstract security model (V-SAT) in Tan et al. (2004a) represents the different security profiles of the system; viewpoints (or profiles) define the requirements and policies for safeguarding the assets, the items of value in the system, against threats. There are different types of security policies including deterrents (configuring safeguards to deter against threats), detectors (that monitor and filter events for say possible intrusion events), repairers (that handle attacks that have occurred or are occurring) and counter-measures (that actively attack the source of the threat itself). Safeguards may have further safeguards to protect them, e.g., private keys used for authentication may require additional access control and confidentiality safeguards. Specifying such viewpoints or profiles are not sufficient—computational models are needed to analyse and configure systems that adhere to them.

The development of a reasoning model provides the means to arrive at decisions related to the security requirements expressed in the profiles. An ontological model (Tan et al., 2004a) related to this reasoning model promotes the interoperability between disparate security systems, providing a framework to support common knowledge across heterogeneous multi-domain boundaries. In developing this model, there was a need to specify an abstract representation determining how the reasoning mechanism or algorithm should work, so that it is not directly dependent on a particular system. On the other hand, an explicit model was developed to ground the theoretical aspects of this model as a proof of concept. Fig. 2 illustrates a schematic view of the policy reasoning model.

Fig. 2 describes how an administrator can dynamically update the externally visible security policies defined in published profiles or viewpoints. Agents can interoperate and reason about these profiles, match safeguard configuration requests and resolve conflicts to instantiate an appropriate common security service configuration. The framework relies on other models such as the common security ontology (meta in Fig. 2) and operational models (Tan et al., 2004a). It is also possible to orchestrate policies into composite profiles covering multiple domains. Fig. 3 specifies the reasoning model of Fig. 2 in more detail.

This model specifies a system for reasoning about security profiles in open environments using mixed, object and meta-level reasoning mechanisms. The reaction to environment conditions (spatial and temporal), static preference rules (general policies) and operational permissions (service determining the executed security requirements) are concerns at the object level.

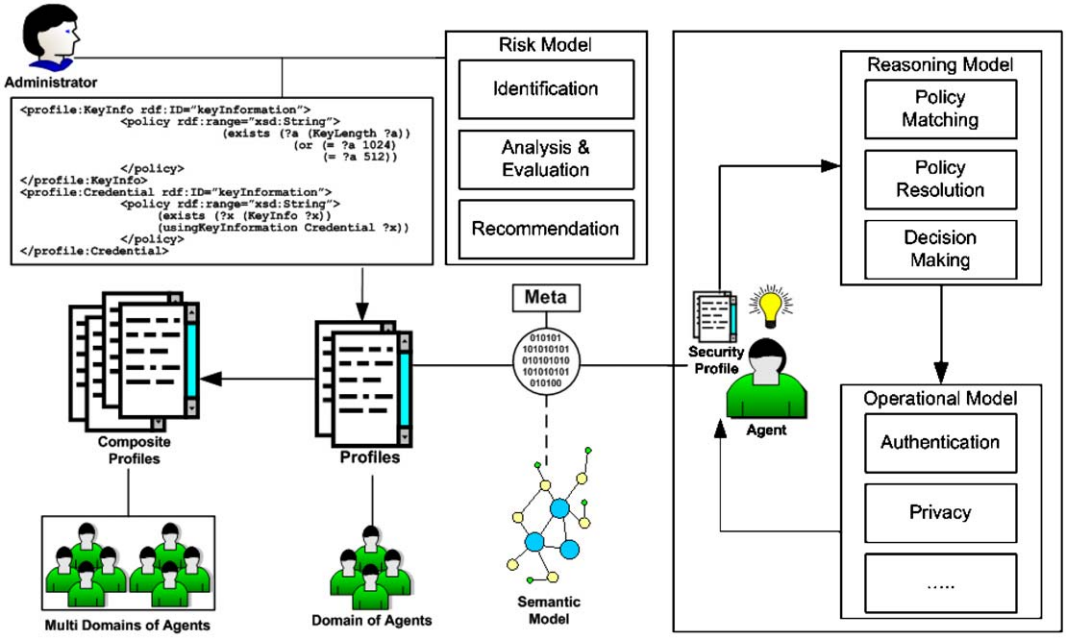


Fig. 2. Framework for open service security.

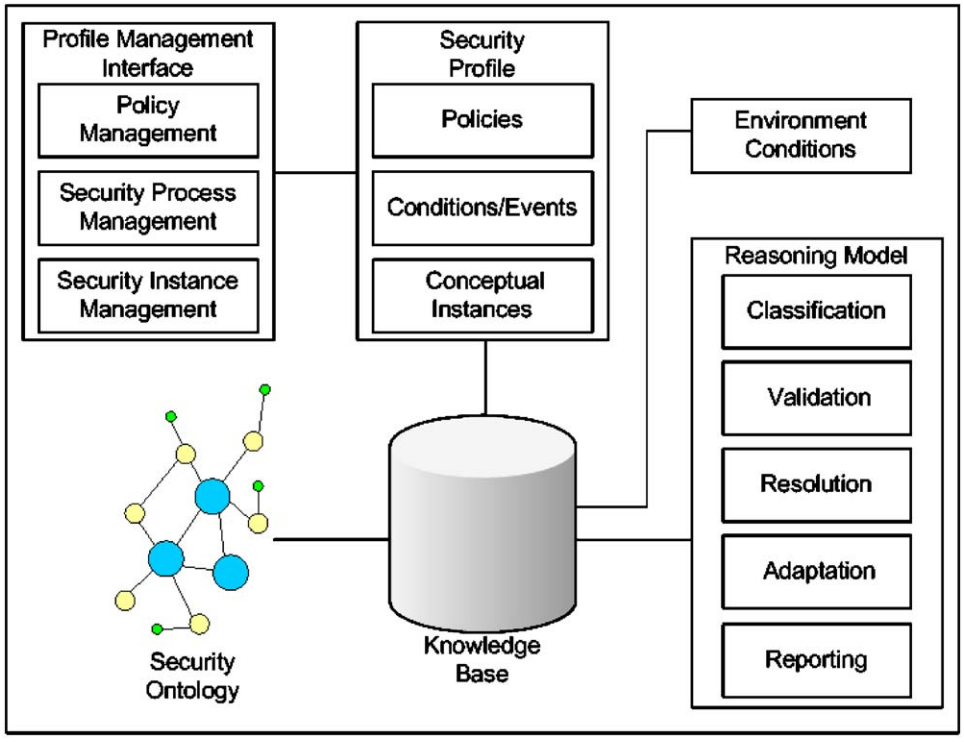


Fig. 3. Profile-based reasoning model.

The common security ontology (domain knowledge) and the security profiles (control knowledge) are used by the reasoning model at the meta-level.

### 3.1. Profile management interface

The profile management interface is a collection of sub-management components that provides an interface for configuring security profiles, policies and instances.

#### 3.1.1. Policy management

There are two main security policy domains: generic security policy domains and service specific policy domains (Sections 4.1.1 and 4.1.2). They are loaded into the knowledge base and serve as policy rule sets. The security policy manager separates the generic security policies from the service specific policies and from application and user preferences. The policy manager interface supports the use of standard declarative semantic languages, such as KIF and RDF, for expressing policies. The limited language-specific expression of policies can be constrained by defining subsets of these languages with their relevant axiomatic semantics. Additionally, the policy expressions can be based upon axioms defined using terms from the abstract security ontology.

#### 3.1.2. Security process management

This defines the security process models within an open service environment. Service description languages provide hooks for interleaving complementary security processes within an application's fundamental services. This interface can provide the methods for defining safeguard models such as authentication, confidentiality or integrity and their required input, output and effect parameters.

#### 3.1.3. Security instance management

The instances, based on the abstract security ontology, represent the actual security configuration supported by the users and service providers. Example instances are described in Tan et al. (2004a).

### 3.2. Security profiles or viewpoints

The profile contains entities that express the safeguards, policies and configuration an user or service provider supports. It provides the mappings between the following models: semantic model (ontology), open service model (service descriptions), and the policy model (constraints and rules). The profile also identifies the various assets one system or domain may wish to protect at various levels of granularity. The granularity exists as a hierarchical nature, where multi-systems can be protected at different levels for example at the service, application domain or multi-domain levels. The profile can be used to manage policies using an *event-condition–action* paradigm in which aspects of the security behaviour can be controlled by a set of rules.

### 3.3. Security ontology

The security ontology defines the security concepts and their relations according to the V-SAT abstract model. Agents and services share the security ontology with the reasoning system. The ontology is loaded with the foundational facts for the security knowledge base. The ontology model defines the semantics for the V-SAT abstract model in more detail, for example, safeguards are defined using the concepts of protocols and credentials.

### 3.4. Knowledge base

The knowledge base (KB) is a repository that maintains the semantic information derived from the security ontology and profiles. The knowledge is a structured representation based on RDF and exists in memory within the Java theorem prover (JTP) reasoner. The KB is a repository storing facts, and it is separate from the reasoning model, which represents the computational cycle of processing rules and facts. The repository manages this data and its relationships, where facts about security are maintained in a semantic network. This allows rules in the form of queries and assertions to support proofs for rational decision making. A reasoning engine is closely related to the KB as part of the reasoning model to execute processes such as validating and discerning facts from the ontology, and executing rules from profiles.

### 3.5. Reasoning model

The reasoning model adopts a set of logic reasoning rules based on the facts in the knowledge base. It can deduce decisions to support the reconfiguration of user security properties. The reasoning model can manage both the policies and the facts of the system to achieve a rational consensus. The consensus is the result of reasoning between profiles originating from different entities and domains, where the steps within the reasoning process are based on a modelling choice that was considered appropriate. The reasoning model in Fig. 3 is defined in detail below, and concrete examples are available in Section 4.

*Classification*: The security ontology is initially loaded into the knowledge base, and followed by various general policy and facts. The classification sub-system is triggered when it receives an invocation for logical reasoning about security profiles. As a result, the classification stage classifies the user policies derived from the profile as incrementing checkpoints to create a data model in the reasoning engine in which policy reasoning can be fired or triggered in a systematic order. This takes the *n-arity* policy (a policy constituting a number of alternative constraints) precedence into consideration (Tan et al., 2004a), where policies are reasoned logically based upon their declared priority.

*Validation*: Generic security policies can be utilised to provide overall integrity checks of user policies. These can provide spatial and temporal validation of policies to draw the need for further reasoning, e.g. if entities exist within a trusted domain, some security requirements can be less important. The purpose of the sub-system is to optimise reasoning requirements based on case-to-case scenarios, for example deciding specific rules to be applied to the reasoning process. The validation process is executed whenever new profiles are loaded but especially when interacting with differing services. Consequently, the validation process also involves the verification of any predicates and axioms expressed in the policies against the core ontology facts and it is used to maintain the integrity and conclusions of the reasoning model (detailed in Section 4.1).

*Resolution*: There may be occasions when policies defined by users or services do not match (they have different requirements). In this case, conflict resolution is required to achieve mediation between entities. For this reason, policy conflict resolution in this model is supported with the use of policy precedence (*n*-arity) specified by the system administrator, where a set of policies are represented as alternating preferences of a user. Not only does this method increase the flexibility of the model to support policy matching, it also requires less overhead compared to policy negotiation.

*Adaptation*: In this process, the logical reasoning of policies is related to sets of facts that are maintained within the knowledge base. These facts are grouped into sets to provide the flexibility of inserting and removing groups of facts, which are related or unrelated to these policies. This configuration is used primarily to associate certain facts with certain policies, thus supporting the reasoning of different security profiles using a single reasoning model instance. Consequently, policies will be fired according to the facts, and if a set of facts does not correspond well with the policies, the use of a different set of facts may be applied.

*Reporting*: This process compiles the result of the reasoning procedure to reach decisions. Example decisions are used as operational instructions of an agent to determine the safeguards and constraint requirements needed to establish the security interoperability between MAMD entities (detail examples in Section 4.3).

## 4. Reification of reasoning model

To make these ideas more comprehensible and appraisable, a scenario is given in Fig. 4. The example describes an open environment setting where different systems publish their services along with them externally

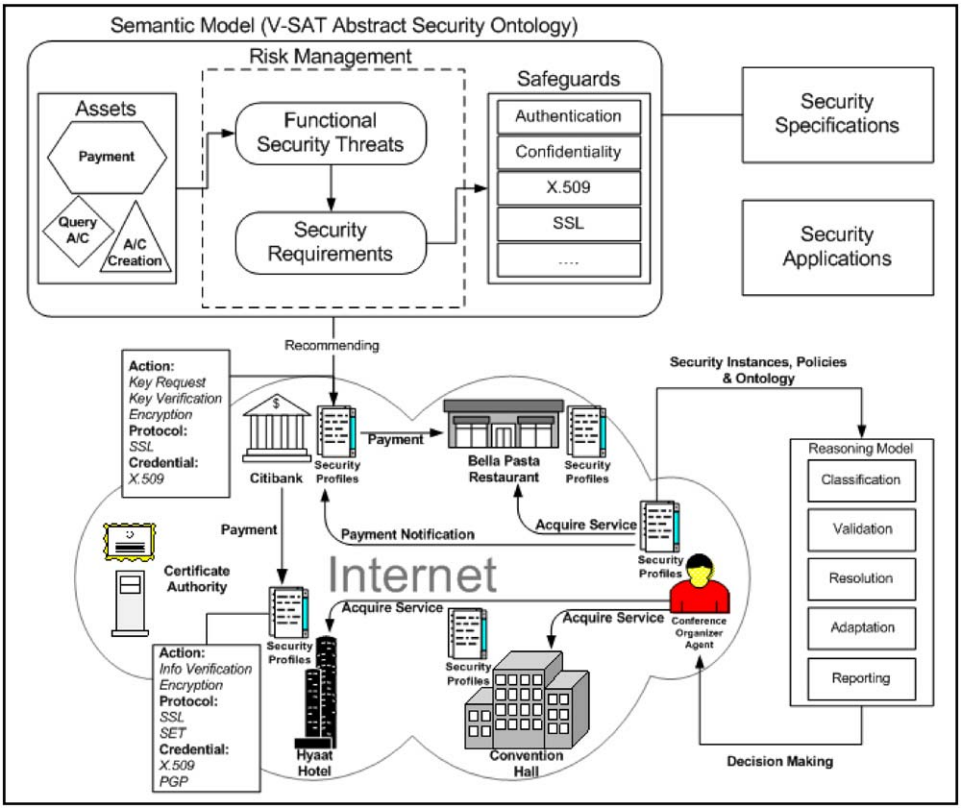


Fig. 4. Service composition scenario.

public security configuration and requirements. These service descriptions have included a detailed service process description of their security choreography or workflow. The security processes are represented as profiles enforcing their respective security configurations (protocols, credentials, and actions) and policies. Some of these configurations include combinations such as SET or SSL, X.509 certificates, and "confidentiality" or "authentication" actions. In the scenario, a "Conference Organiser Agent" (COA) wishes to organise an event and interacts with services such as a convention hall, a restaurant and a hotel. Initially, an agent discovers these services through directories and discerns the security choreography and profile. Through the profile description and policies governing these security instances, an agent is able to reason about the profiles with the aid of the holistic security ontology, hence capturing and understanding the concepts and processes needed to support interoperability between disparate services. Eventually payment is made through the banking service and a conference event is organised. The security profiles represented by each service are created by the semantic and risk models in Tan et al. (2004b) briefly illustrated in Fig. 4. The following figure outlines the reification of the model for the given scenario.

The reification of the reasoning model in this section accounts for an explicit description of the abstract model. The model refers to five core processes (Section 3.5) related to the reasoning model. A reification of these processes in accordance with the scenario in Fig. 4 that describes the application of reasoning rules, and a summary of these processes expressed as an algorithm is given in the following sections.

### 4.1. Classification and validation processes

The security ontology comprising the common KB is loaded as sub-processes with associated checkpoints (partitions) into the KB. Thus the KB can be partitioned to enable the system to dynamically load or reinstate previous facts in the KB on runtime. This is important for handling multiple security profiles, and to ensure its integrity and consistency when importing new facts and to be able to revert back to previous facts using checkpoints (Section 4.3). The security profiles belonging to the bank and other services are systematically loaded as part of the KB by the COA. Subsequently, the profiles and environment conditions undergo the validation process based upon different types of generic security policies and service domain specific policies of the service.

#### 4.1.1. General security policies

General security policies include common deterrence rules for safeguarding assets but may also include threat

detection and counter measures. These rules are specified dependent upon individual domains, and it reflects the common configurations of domains or services. Thus these policies include *temporal and spatial policies, safeguard–threat policies*, and *decision making policies*.

Temporal and spatial policies relate to environment conditions. Inputs are received through the monitoring of spatial and temporal information from the local domain knowledge. In a domain environment such as the bank service (Fig. 4), all processes are constantly kept secured. On the other hand, chains of "Hyaat" hotels can interact with entities from within the same domain as a trusted environment from 0900 to 1700 h, but not at other times (Table 2). According to two environment parameters such as time and location, the following policy instance is defined:

- $\forall t\ \text{Time}(t) \wedge ((t < 9) \vee (t \geqslant 17)) \wedge$
  $\text{Location}(\text{HyaatDomain})$
  $\Rightarrow \exists x\ \text{Threat}(x) \wedge$
  $\text{type}(x\ \text{Repudiation})$
  *Before 0900 or after 1700 and in a "Bella Pasta Domain" implies the existence of a "repudiation" threat*.

Subsequently, safeguard–threat policies specify which type of safeguard is needed to protect against certain threats (Table 2), for example:

- $\forall x\ \text{Threat}(x) \wedge \text{type}(x\ \text{Repudiation})$
  $\Rightarrow \text{need}(\text{Integrity})$
  *For all* x *of type "repudiation," it implies "integrity" safeguard is needed*.

Decision making policies deduce decisions from the general security policies (Section 4.1.1). Derived decisions are further reasoned to determine if atomic attributes (constraints) supported by both the service provider and COA match. The matching method is discussed in Section 4.1.2, but if the instances do not match atomically, the policy negotiation method will be fired (Section 4.2) to attempt the resolution between conflicting policy constraints. "Match" refers to the matching of policy constraints defined by service security instances to the user security requirements (Table 3). The following is an example policy:

- $\forall x\ \text{need}(\text{Safeguard Integrity}) \wedge$
  $\text{type}(x\ \text{Integrity}) \wedge$
  $x\ \text{UserPolicies Match}(x\ \text{UserPolicies})$
  $\Rightarrow \text{negotiate}\ (x\ \text{UserPolicies})$
  *For all* x *that is of type integrity and if integrity is needed as safeguard, and* x *does not match with requirement, it implies the need for UserPolicies about integrity*.

### 4.1.2. Service policies

Service policies are the service requirement policies. These policies define the detailed constraints of service instances. For example if more than one protocol is supported, the policies provide services with the configurability to specify precise requirements (Table 3). Policies can also define alternate constraints to provide better configurability and heterogeneity. The following rules describe some examples interlinking the safeguards with the protocols, credentials and actions that are being used.

- *For all* x *and* y *where* x *of type Password and* y *of type Symmetric Key, it implies using* x *as a credential and* y *as a credential.*
  $\forall xy\ \text{type}(x\ \text{Password}) \wedge$
  $\text{type}(y\ \text{Symmetric Key})$
  $\Rightarrow \text{Credential}\ (x) \wedge \text{Credential}\ (y)$
- *For all* a, x, y, z *where* a *of type Authentication and* x *of type Key Distribution and* y *of type SSL and* z *of type Password and* a *has credential* z *and* a *has protocol* y *and* a *has action* x*, it implies UserPolicies*

Table 2
Example of spatial and temporal conditions for capturing associated threats and safeguards

| Spatial and temporal conditions | Threats | Safeguards |
|---|---|---|
| Bank service: Location = internal/external, Time = any | Eavesdropping, repudiation, masquerade | Authentication, confidentiality, integrity |
| Hyaat hotel: location = internal, time = 0900–700 h | Nil | Nil |
| Hyaat hotel: location = internal, time = 1700–0900 h | Repudiation and masquerade | Authentication, integrity |

Table 3
Example of results determining what actions and instances to be invoked by the operational model

| Results | Actions | Instances |
|---|---|---|
| Exact match | Authentication (credential exchange) | Credentials: X509 Protocols: SSL |
| Resolution match | Confidentiality (encrypting personal details) | Credentials: X509 (DSA-SHA1) |

*match* a.
∀ a, x, y, z type(a Authentication) ∧
type(x KeyDistribution) ∧
type(y SSL) ∧ type(z Password) ∧
hasCredential(a z) ∧
hasProtocol(a y) ∧ hasAction(a x)
⇒ Match(UserPolicies a)

Fig. 5 defines the COA and services' security instances in preceding order as they are listed. Service policies govern the utilisation or preference of instances to meet different requirements. The reasoning model derives three types of results (*exact match, resolution match,* or *mismatch*) by utilising instances and policies to determine their compatibility. If a mismatch is discovered in the first instance, the model attempts to resolve the problem to derive a resolution match or a final mismatch result.

### 4.2. Policy conflict resolution

Policy conflicts can occur when security requirements of different parties and domains do not share similar constraints (restriction condition of a given instance–in bold below). As a result, policy resolution methods can resolve most conflicts appearing in these scenarios. The following describes an example from Fig. 5 of two policies defining the algorithm and protocol instances using a disjunctive operator. These instances can be termed and prioritised as Policy1 and Policy2. The following is a resolution example that is linked to the algorithm specified in Table 4 (5):

Policy1:
(exists (?a (Algorithm ?a))
(or (= ?a RSA-SHA1) (= ?a DSA-SHA1))
Policy2:
(exists (?p (Protocol ?p))
(or (= ?p SET) (= ?p SSL))
Their priorities are decided as,
constraint_1 > constraint_2 > constraint_n.

The algorithm holds a number of policies and constraints and subsequently deduces if each constraint of the service profile (SP) matches with the user instances (UInst). Hence, the policies are validated with the instance's constraint of the service. If they do not match, the resolution method will handle the subsequent policy until a final resolution is achieved based on the priorities (*n*-arity).

### 4.3. Adaptation and reporting processes

The adaptation process allows the KB to manage the facts and rules loaded into the system in terms of sets

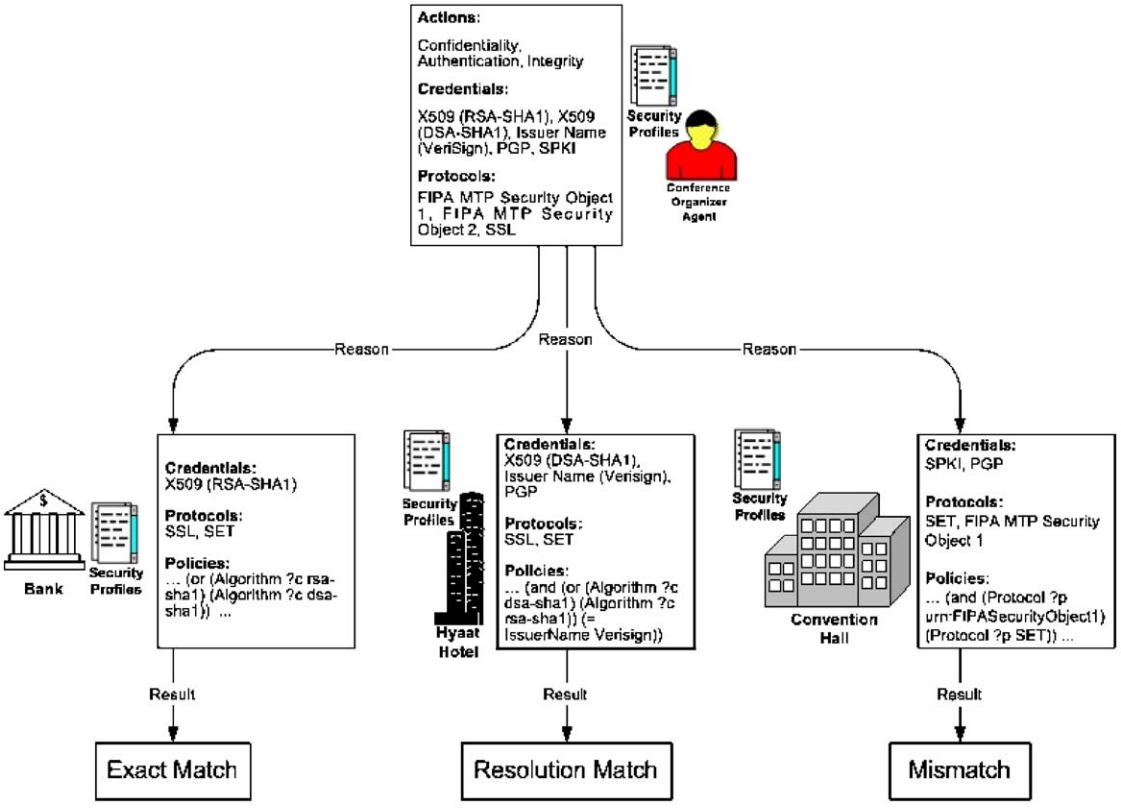


Fig. 5. Example of COA instances, services' instances and policies with reasoning results.

Table 4
Formal specification of reasoning algorithm

| | |
|---|---|
| $\mathrm{KB} = \bigcup_{K_j \in \mathrm{KB}} K_j$ | (1) |
| iter $R^k = (\exists x$: temporal; $\exists y$: spatial $\mid \neg T[t_n] \bullet (x \in T \wedge y \in S) \wedge (x \geqslant T[t_n] \wedge x \leqslant T[t_n]))$ | (2) |
| $R_1 \leftarrow \sum_{j=0}^{n} \mathrm{TH}[\mathrm{th}_j] = (\exists z:$ threat$\mid \mathrm{TH}[\mathrm{th}_n] \bullet z \in \mathrm{TH}\ S[\mathrm{sg}_n])$ | (3) |
| $R = (\exists(\mathrm{inst}, p); \exists \mathrm{sg}:$ safeguard$\mid$match $\bullet ((\mathrm{sg} \in \mathrm{SP}) \wedge (\mathrm{inst} \in \mathrm{SInst} \wedge p \in SPol) \in \mathrm{SP}) \wedge$ MATCHING) | (4) |
| $\mathrm{MATCHING} \leftarrow \sum_{j=0}^{n} \mathrm{Ans}_j = (\exists(\mathrm{inst}, p) \mid \mathrm{Ans} \bullet (\mathrm{inst}, p) \in \mathrm{SP} \equiv ((\mathrm{inst}, p) \wedge \mathrm{UInst}))$ | |
| $R \leftarrow \sum_{j=0}^{n} C[c_j](\exists c:$ constraint; $\exists \mathrm{inst}:$ instance$\mid C \bullet ((\mid C \mid \in P) \in \mathrm{SP}) \wedge (c \equiv (\mathrm{inst} \in \mathrm{UInst})))$ | (5) |
| $R \leftarrow R \cup (C[c_n] > C[c_{n+1}])$ | (6) |

and to define states called checkpoints before each set is loaded. Defining such checkpoints within the KB permits the system to revert back to previous states. It is also easy to unload unused profiles ($x$) and load new profiles ($y$)—without checkpoint control there might be a need to unload and reload the entire KB. This provides better management of resources and improves performance (Section 5.2). Consequently, the reporting process computes these results (Fig. 5) to derive a rational decision for either the user or system to instantiate the operational model. Similarly like an expert system, the exact match or resolution match result provides an output indicating what operational decisions should be executed by the system to bridge the security interoperability gap (Table 3).

### 4.4. Reasoning algorithm

In the following, the computation process of the algorithm taking into account the model and reification explained in Sections 3 and 4 is defined based on Table 4:

1. Ontology, general rules, user and service profiles are loaded into the KB as the Union of $K_j$.
2. 'Environment threat detection policy' determines the need for security based on temporal and spatial security requirements (environment inputs). If unspecified, the reasoning process continues to deduce other security requirements. $R$ is an iterated (iter) result "such that" (|) $\neg T[t_n]$ (*temporal set of elements*) is satisfied where (●) the conditions of $x$ within the boundary of $T[t_n]$ are not met iteratively ($T$—*temporal,* and $S$—*spatial*). On the contrary, if $\neg T[t_n]$ is not satisfied where the conditions are true then system ends the process, affirming that the environment is trusted. This can be referred to the rules specified in Section 4.1.1 indicating if variables within the spatial and temporal constraints are subjected to an un-trusted environment, leading to the need for security safeguards.
3. If threats exist, the reasoner checks for all associated threats to determine the required safeguards. This loop operation captures all the relevant set of threat elements ($\mathrm{TH}[\mathrm{th}_n]$) and its associated set of safeguard elements ($S[\mathrm{sg}_n]$).
4. The safeguard associated with the service profile has policies and instances matched against the user profile instances. If these profiles match, the reasoning process ends, and a decision is specified. The algorithm seeks to match answers (Ans) via a loop and assigns them to *MATCHING*. Consequently to determine the result ($R$) by satisfying *match*, the association of the SP, service instances (SInst), and service policies (SPol) *is validated*.
5. If they do not match, the 'policy resolution' process determines if the user profiles instances can support the alternative service policy constraints. The results are stored and processed to determine the precedence of alternative policy constraints supported by the service. The loop seeks for all satisfied set of constraints ($C$) where the Cartesian of $C$ are policies ($P$) belonging to SP and each constraint element ($c$) is equivalent to the user instances. The results are stored in an order and their priority is determined by precedence.
6. A result is presented to the user amalgamating the outputs obtained by the reasoning process in the form of the decision making process. This provides the necessary information to the operational model.

## 5. Discussion and critical analysis

The policy-based model has been used over a range of Agentcities agent services such as market places, event organisers, e-banking agent systems and a number of M.Sc. projects (Agentcities). The layered security framework provides an account of the models defining the framework (Tan et al., 2004b). The algorithm implemented in this system supports a combination of

processes that classify, validate, resolve and can decide rationally about profiles. The algorithm has been tested with profiles available in Tan (2003) for cases when the security requirements of both users and services can be cross-matched utilising the reasoning algorithm. An ontology representing abstract and concrete concepts derived from IETF, Oasis and W3C security standards, and an API supporting security functionalities have been developed. A discussion and evaluation of the framework in terms of its performance is given below.

### 5.1. Evaluation of the framework

A holistic abstract yet explicit framework for securing open services in MAMD environment promotes security interoperability between multiple services. This V-SAT framework is motivated by conflicting security instances and policies between different islands of service implementations. In retrospect, the framework possesses one or more characteristics of autonomic computing, to support its application in open security interoperability for service composition scenarios. In the first instance, the framework supports the use of security profiles as the means for permitting services to explicitly specify security instances, policies and processes along with existing service descriptions. These features allow the reconfiguration of security requirements and policies with minimal human intervention; a system administrator may alter system operations without affecting its underlying implementation. Subsequently, the framework supports system protection by utilising a semantic and risk-based approach. This is a step towards developing "safety" in open environments where a number of core security configurations are recommended to a system using quantitative and qualitative risk management methods (Tan et al., 2004b), thus decreasing the possibility of over-configuring a system that may lead to a bad security configuration. The reasoning model supports the analysis of security profiles between systems by classifying facts and rules, validating security requirements, and selecting suitable recommendations for resolving conflicts and mediating security interoperability. Consequently, the framework supports optimisation through the mobilisation of constantly changing resources and requirements of open systems by adapting to new facts and rules when interacting with multiple services. This is achieved by maintaining a knowledge-base of core facts and managing changing ones at different checkpoint levels without the need for reloading the same ontologies when interacting with different services.

The approach of evolving an infrastructure in the direction of autonomic computing is based on 5 levels that run from manual to autonomic: *basic, managed, predictive, adaptive* and *autonomic* (Ganek and Corbi, 2003). In comparison to these levels, the holistic security framework is closely related to features defined in the predictive, adaptive and autonomic levels, where the framework exercises information correlations and recommendations, system management through security process logic and policies, and supports the harmonisation of different instantiations to advocate security interoperability. The risk-management model (Tan et al., 2004b) within the framework supports the identification, analysis and evaluation where security vulnerabilities can be measured to offer quantitative security recommendations. The V-SAT model represents an ontological model for specifying security processes, instances and policies at a service description level where businesses can manage the operational logic of their system in a high-level approach. The reasoning model for this purpose enables the analysis of multiple security profiles to decide if a suitable level of security interoperability between heterogeneous systems is achievable.

#### 5.1.1. Meta-reasoning and autonomic computing

In relation to mixed level meta-reasoning, the structured perspective of the system is referred as domain and control knowledge. The former defines the ontology, and the latter defines the security profile and processes. The structured perspective supports a standpoint whereby object level instantiations are represented at a meta-level advocating abstract management of a system where components are loosely coupled. This intrinsic characteristic is complementary to the notion of self-management within autonomic computing where the meta-level allows a decentralised method for managing knowledge outside implemented systems, thus permitting the utilisation and sharing of ontologies between disparate applications. Therefore, meta-level architectures are suitable for heterogeneous environments where purely object level ones fail at providing conceptual knowledge for systems to dynamically interoperate securely.

#### 5.1.2. Modelling choice

The motivation for developing this model is to try to reuse applicable current security models via a holistic upper security ontology to support agent security interoperability for web services and beyond. The security model was developed in this way to avoid designing the system from scratch in a proprietary way. If agent systems are to interoperate within the heterogeneous business environment, it is useful if MAMD models can interoperate with current best-practice business security models.

The (V-SAT) model was introduced in this manner where profiles, safeguards, threats, and assets are used as the basis of an abstract model for capturing stakeholders in MAMD. This abstraction allows heterogeneous

systems to model and capture their entities in an abstract way that can be strapped onto an explicit model. In this way, numerous systems can rely upon this abstraction to determine the relation between stakeholders of their own and other systems. Interoperability is more achievable because an explicit model is defined that is generic and not dependent upon a specific application domain. Security can also be modelled at different levels ranging from hard, hybrid to soft security.

### 5.1.3. Need for an explicit security model

A model of security is needed in order to understand and clarify the security properties a system should possess and to state them explicitly before embarking on development. This model of security is often referred to as the security policy. It defines a specification of the protection goals, and the security requirements of a system. This is driven by an understanding of the threats to the systems assets (the items of value in the system) and the safeguards. This will help explain and communicate the use of security to the different stakeholders of the system such as end-users, application developers and system maintainers. The argument that the security configuration should not be revealed because advertising how the system is protected enables attackers to gain useful information to gain unauthorised access, the so-called security by obscurity, gives systems a false sense of security. Security by obscurity may make the initial attacks harder for the adversary but this may hide weaknesses that an open peer review might have revealed.

There is a trade-of in analysing MAMD system security in such an abstract way. The advantages of this kind of abstract reference model include being insulated from popular particular technological security models that may become disused or superseded and being able to support heterogeneous application security requirements. The disadvantage is that an abstract model may appear to be too abstract, complex and flexible to be used to specify concrete MAMD security service requirements in practice. In order to keep the advantages and to minimise the disadvantages, a profile-based approach is used to map an abstract common view of security to particular application-oriented reifications of the model. An abstract security (policy) model is a specification of the protection goals of the system. Technology-driven protection mechanisms may protect the wrong or incomplete things because a security policy has been misunderstood. Less formal representations, such as graphical models of policies often lead to the use of implicit constraints later in development and greater complexity in managing policies.

## 5.2. Evaluation of performance

An evaluation of the performance of the reasoning model developed using JTP, and its ontologies and profiles written using DAML+OIL has been undertaken. The policy language is based upon Knowledge Interchange Format (KIF). The system is loosely coupled and can be easily instantiated with existing semantic based technologies such as web or agent services (Tan et al., 2004a). The model can be bootstrapped either by a GUI or simply instantiating Java method calls to the reasoning engine into existing applications, where ontology and profiles are loaded through URN(s). Some key issues for practical reasoning models are their performance and scalability for MAMD environments. The performance of loading facts and rules into the JTP is given in Fig. 6.

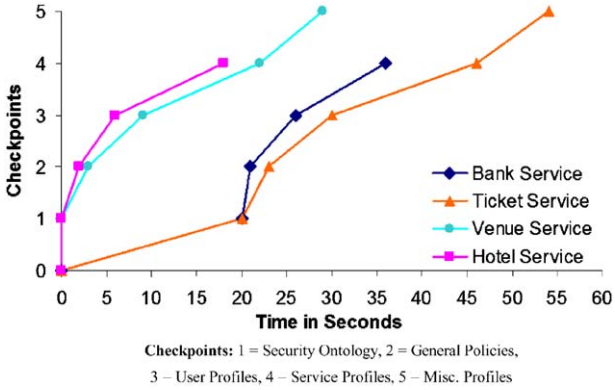


Fig. 6. Performance of loading facts and rules into JTP.

As shown in Fig. 6, the performance of the reasoning system was tested and it approximately takes around 36 s to completely establish its required KB on a 1.5 GHz Pentium 4 notebook computer with 256 MB of memory. The reasoning time is almost negligible but the computation complexity of loading facts and policies can be significant. The core ontology can be loaded in 20 s for around 70 facts and 60 s for around 250 facts. This is largely dependent on the size of the ontology. Therefore, the reasoner is partitioned into various checkpoints where knowledge can be loaded at different intervals. Hence increasing the system's performance where it can load or unload knowledge dynamically. In Fig. 6, the four types of services are grouped into two composite services "hotel and venue services" and "bank and ticket services". N.B. The Hotel and Venue services do not need to load the knowledge for checkpoint 1 if the bank and ticket services have already loaded their KBs. The JTP developers are currently working on performance improvements, which may affect these results.

## 6. Conclusion and future work

Security issues arising from open systems can be difficult to manage and support. There exist numerous

security requirements specifications that can result in end-to-end security interoperability problems. Resolving this issue requires the use of systems that are able to dynamically validate security requirements. A policy-based reasoning model is developed to address this problem in which systems can be controlled at a meta-level without changing their underlying implementation. The feasibility of this model and an evaluation has been presented.

In the future, there are plans to improve the model by simulating more dynamic environments where changing threats can constantly attack the system, and to monitor how the system behaves under these changing conditions. Consequently, this could promote a model that can heal itself against attacks when a system is compromised or if it fails.

## References

Agentcities.RTD. Global Agent Test-bed, http://www.agentcities.net/

Bantz, D.F., Bisdikian, C., Challener, C., Karidis, J., Mastrianni, S., Mohindra, A., Shea, D., Vanover, M., 2003. Autonomic personal computing. IBM Systems Journal 42 (1), 165–176.

Blaze, M., Feigenbaum, J., Lacy, J., 1996. Role of trust management in distributed systems security. IEEE Conference on Security and Privacy, Oakland, USA, pp. 164–173.

Bradshaw, J., Dutfield, S., Benoit, P., Woolley, J.D., 1997. KAoS: toward an industrial-strength generic agent architecture. In: Bradshaw, J.M. (Ed.), Software Agents. AAAI/MIT Press, Cambridge, USA, pp. 375–418.

Chess, D.M., Palmer, C.C., White, S.R., 2003. Security in an autonomic computing environment. IBM Systems Journal 42 (1), 107–118.

Chomicki, J., Lobo, J., 2001. Monitors for history-based policies. In: Sloman, M., Lobo, J., Lupu, E. (Eds.), Policies for Distributed Systems and Networks. Springer, Berlin, pp. 57–72.

Corradi, A., Dulay, N., Montanari, R., Stefanelli, C., 2001. Policy-driven management of agent systems. In: Sloman, M., Lobo, J., Lupu, E. (Eds.), Policies for Distributed Systems and Networks. Springer, Berlin, pp. 214–229.

Damianou, N., Dulay, N., Lupu, E., Sloman, M., 2001. The ponder policy specification language. In: Sloman, M., Lobo, J., Lupu, E. (Eds.), Policies for Distributed Systems and Networks. Springer, Berlin, pp. 18–38.

Department of Defence (DoD), 1985. Trusted Computer System Evaluation Criteria, DOD 5200.28STD.

Foster, I., Kesselman, C., Tuecke, S., 2001. The anatomy of the grid: enabling scalable virtual organizations. International Journal of Supercomputer Applications 15 (3), 200–222.

Ganek, A., Corbi, T.A., 2003. Autonomic personal computing. IBM Systems Journal 42 (1), 5–18.

Grandison, T., Sloman, M., 2000. A survey of trust in internet applications. IEEE Communications Surveys and Tutorials 3 (4), 2–16.

Haas, R., Droz, P., Stiller, B., 2003. Autonomic service deployment in networks. IBM Systems Journal 42 (1), 150–164.

Harmelen, F., 1989. A classification of meta-level architectures. In: Jackson, P., Reichgelt, H., Harmelen, F. (Eds.), Logic Based Knowledge Representation. MIT Press, Cambridge, USA, pp. 13–35.

Herzberg, A., Mass, Y., Michaeli, J., Ravid, Y., 2000. Access control meets public key infrastructure, Or: assigning roles to strangers. IEEE Symposium on Security and Privacy, California, USA. May, pp. 2–14.

Horn, P. Autonomic computing: IBM's perspective on the state of information technology, IBM Corporation, http://www.research.-ibm.com/autonomic/manifesto/autonomic_computing.pdf.

Kaijser, P., Parker, T., Pinkas, D., 1994. SESAME: the solution to security for open distributed systems. Computer Communications 17 (7), 501–518.

Lobo, J., Bhatia, R., Naqvi, S., 1999. A policy description language. Proceedings of the AAAI, Orlando, USA, pp. 291–298.

Lymberopoulos, L., Lupu, E., Sloman, M., 2002. An adaptive policy based management framework for differentiated services networks. Workshop on Policies for Distributed Systems and Networks, California, USA, pp. 147–158.

Marshall, I.W., Mckee, P., 2001. A policy based management architecture for large-scale active communication systems. In: Sloman, M., Lobo, J., Lupu, E. (Eds.), Policies for Distributed Systems and Networks. Springer, Berlin, pp. 202–213.

Miller, J., Resnick, P., Singer, D., PICS Rating services and rating systems, W3C, http://www.w3c.org/TR/REC-PICS-services.

Nomura, Y., Chugo, A., Adachi, M., 1999. A policy-based networking architecture for enterprise networks. IEEE International Conference on Communication, vol. 1, No. 6–10, Vancouver, Canada, pp. 636–640.

Tan, J.J., 2003. Adaptive Management and Interoperability for Securing Semantic Open Services, Queen Mary, University of London, http://agents.elec.qmul.ac.uk/agentcities/security/open/

Tan, J.J., Poslad, S., Titkov, L., 2004a. An ontological approach to harmonising security models for open services. Fourth International Symposium From Agent Theory to Agent Implementation (AT2AI), Vienna, Austria, pp. 594–599.

Tan, J.J., Poslad, S., Titkov, L., 2004b. A semantic approach to harmonising security models for open services. Journal of Applied Artificial Intelligence, Special Issue on From Agent Theory to Agent Implementation, to appear.

Yialelis, N., Sloman, M., 1996. A security framework supporting domain-based access control in distributed systems. Symposium on Network and Distributed System Security, San Diego, USA, pp. 26–39.

**Juan Jim Tan** is currently a research student undertaking a Ph.D. degree at the Department of Electronic Engineering in Queen Mary, University of London and part of his Ph.D. has contributed to the EU Agentcities.RTD project where he has worked in the area of security and interoperability in open service environments. His research interests include: the semantic web, adaptive management, and security interoperability for distributed systems. He received his M.Sc. degree with distinction in e-Commerce Engineering from Queen Mary, University of London.

**Dr. Stefan Poslad** is a lecturer at Queen Mary, University of London. He has worked on several collaborative distributed system projects in the area of: developing mobile user tourism services, integrating environmental data and on a global agent test-bed. He chairs the FIPA agent standards forum security Technical committee and is on its Board of directors. He is an active member of agent, Trust and ubiquitous system thematic networks and a co-organiser of related workshops. His interests include: security, trust and privacy models for open distributed services; open distributed services models based upon the Semantic Web and Intelligent agents, and Ubiquitous computing.