Model-driven Simulation for Cross-domain Policy Enforcement

Zhengping Wu, Lifeng Wang
Department of Computer Science and Engineering
University of Bridgeport
{zhengpiw,lifengw}@bridgeport.edu

***Abstract*—This paper proposes an enforcement architecture and develop a simulation framework for cross-domain policy enforcement. The entire simulation environment is used to solve the problem of enforcing policies across domain boundaries when permanent or temporary collaborations have to span over multiple domains. In reality, different systems from different organizations or domains have very different high-level policy representations and various low-level enforcement mechanisms, such as high-level security policies, privacy configurations, and low-level system calls (services). To make sure the compatibility and enforceability of one policy set in another domain, a simulation environment is needed before actual policy deployment and code development. The framework developed in this simulation environment can also be used to generate policy enforcement code directly for permanent integrations or temporary interactions. This framework provides various functions to enforce policies automatically or semi-automatically across domains as by-products. A case study in healthcare information systems confirms the advantages of these new functions and facilities in this simulation environment.**

***Index Terms*—**model-driven simulation, policy enforcement, policy modeling, cross-domain enforcement.

I. INTRODUCTION

olicy-based management is an administrative approach to Psimplify the management of a given endeavor by establishing policies to deal with situations that are likely to occur. Policies are operating rules that can be referred as means of maintaining order, security, consistency, or other ways of successfully furthering a goal or mission. Different communities, organizations and domains have their different standards to define policies and policy execution infrastructures to enforce their policies. These policies could be defined by any types of policy languages such as WS-Policy and XACML [3]. Low-level enforcement mechanisms could be very different from system to system. So it is hard to enforce a policy across domain boundaries or over multiple domains. Before applying policies across domain boundaries, it is desirable to know which policies can be supported by other domains' enforcement mechanisms, which are partially supported, and which are not supported. A simulation of cross-domain policy enforcement can help system administrators decide not only the applicability of policies at foreign domains but also the workload to support policies from foreign domains. In this paper, we propose and implement an innovative simulation environment using semantic modeling and translation for policy enforcement across domain boundaries. As a byproduct, this proposed enforcement framework also automatically generates a part or all enforcement code if elements in a policy model can find their corresponding low-level enforcement mechanisms, which cab reduce developers' workload

In the proposed simulation environment, the entire policy-based management architecture is divided into three levels, which can be represented by high-level policy language models, intermediate-level processing models, and low-level policy enforcement models respectively. These three types of models are defined by a semantic language-Web Ontology Language (OWL). The simulation environment can accommodate any types of high-level policy languages; system administrators can easily introduce a foreign policy when a new collaboration is created; and our semantic mapping and translation throughout the enforcement framework is flexible. This environment can simulate policy enforcement for temporary co-operations between or permanent integration of applications and systems from multiple domains.

II. POLICIES FROM MULTIPLE DOMAINS

Policies are operating rules that can maintain order, security, consistency, or other ways of successfully furthering goals or missions in information systems. In healthcare applications, HIPAA requires certain operation policies and privacy policies to protect the healthcare information of patients. Since there are more and more collaborations and communications between domains, cross-domain policy enforcement is a necessary component in these domains' information systems. But in most cases, these domains use different high-level policy languages to define their policies, and these particular policies are executed on their own policy enforcement platforms. Once the collaborations or communications are needed by two "stranger" domains, technical departments from these two domains have to work together to evaluate whether is this possible to make their systems work together, and how much work is needed to establish the collaborations or communications. It is a complex procedure for both participant domains. Thus, a simulation environment can help evaluate this possibility and give an approximate workload for the implementation of collaborations or communications. As an

COLLABORATECOM 2010, October 9-12, Chicago, USA

DOI 10.4108/icst.trustcol.2010.5

integral part of collaboration or communication control, a good simulation environment can also sort out the odds in potential cross-domain policy enforcement and execution. For example, in social networking sites (one social networking site is an independent domain), privacy protection rules can be formally expressed in policies. When people join social networking sites, they begin with creating a profile, and then making connections with existing friends as well as new friends they meet through these sites. A profile is a list of attributes associated with an identification, which includes your real name (or a pseudonym), photographs, birthday, hometown, religion, ethnicity, and personal interest. This list may also contain a person's hobby, interest and other types of information, which may be considered as privacy, such as current and previous schools, employers, drinking habits, and sexual orientation [6,7]. As we know most of existing social networking sites have privacy configurations based on their own enforcement mechanisms. All targets of access control can be simply called "objects" here, such as profiles, photos, videos, daily logs. People who desire to visit these objects can be simply called "subjects". Below, we use the privacy configurations from three major social networking sites as examples to illustrate common points and differences in real policies from multiple domains.

Figure 1. Linkedin Privacy Configuration

For privacy protection, Facebook allows users to define access control policies to protect their "Profile", "Basic Info", "Personal Info", "Status and Links", "Photos Tagged of You", "Videos Tagged of You", "Friends", "Wall Posts", "Education Info", and "Work Info" through a privacy configuration interface. These accessible resources are objects in privacy policies. The access groups include "Everyone", "My Networks and Friends", "Friends of Friends", and "Only Friends." All these access groups are subjects in privacy policies. Myspace allows users to define a similar set of access control policies. Being different from the previous two social networking sites, Linkedin supports privacy control policies for "Research Surveys", "Connections Browse", "Profile Views", "Viewing Profile Photos", "Profile and Status Updates", "Service Provider Directory", "NYTimes.com Customization", "Partner Advertising", and "Authorized Applications" through its configuration interface as well. Compared with Facebook and Myspace, Linkedin's major user groups consist of business and professional people. Most of these users are small and medium enterprises' employees, consultants and sales personnel. Special privacy settings for research surveys and partner advertising can be considered as "special objects" in privacy policy definition. Other objects are very similar to Facebook and Myspace. Figure 1 illustrates Linkedin privacy configuration interface.

In these social networking sites, most privacy settings are similar such as settings for online status, profile, friends, and photos, because the common privacy control rules are quite similar from site to site. But some social networking sites also have distinguished features in privacy policies, such as research surveys' privacy control in Linkedin. This situation can be found in not only social networking sites but other enterprise systems as well, such as healthcare systems. For example, two hospitals have their specific operation policies enforced on different enforcement mechanisms, but their security and privacy policies follow the same set of HIPAA conformant rules. When two social networking sites or two healthcare domains need to communicate or cooperate with each other, they have to rebuild or reconfigure their system to make sure these activities are consistent with their own and partners' policies. When there are hundreds of communication or collaboration partners, we have to rebuild or reconfigure the systems for hundreds of times, which is totally infeasible. Sometimes, the communication or cooperation is temporary, which make system rebuilding or reconfiguration even impossible. So, we need a simulation environment to find out whether every rule in communication or collaboration policies is enforceable over all partner domains. This simulation environment can also tell how many policies used in local domain are similar to those in partner domains. After such simulation, a system administrator can decide whether a system rebuilding is needed or partner domains' policies can be enforced on current execution platform. We propose a new enforcement hierarchy in this paper to provide this simulation, which can not only help make this decision but also generate most enforcement code for a partner domain's policies automatically if the decision is feasible. Detailed information for this enforcement hierarchy and a formal description of the enforcement architecture are provided in section 3 and 4.

III. ENFORCEMENT HIERARCHY

High-level policy languages are easy for users to define policy rules directly, which include natural languages and formal policy languages. They are intuitive for readers to understand policy rules. However, these policy languages tend to be more and more complex with the development of mathematics-derived languages and logic-based languages, such as role-based access control languages, Keynote policy language, and General Access Control Language [1]. One of the most important aspects for interconnections between social

network websites are their agreements of privacy policies. These agreements reflect standard policies defined by social network administrators and designers in high-level policy languages such as XACML. Sometimes, these policies cannot have one-on-one correspondence to low-level enforcement mechanisms in social network execution platforms. So we need to add something between high-level policy languages and low-level mechanisms to resolve the discrepancy. We introduce an intermediate-level mapping and translation layer to connect the two levels. The hierarchy of enforcement is illustrated below.

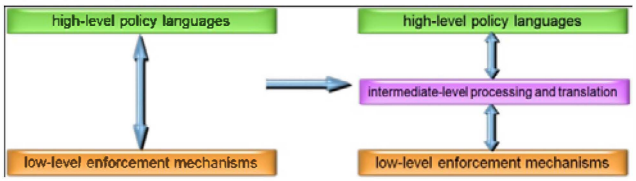

Figure 2. Policy Enforcement Hierarchy

The intermediate-level mapping and translation mechanisms and the corresponding models used in these mechanisms must be flexible enough to bridge the semantic gap between high-level policies and low-level mechanisms in order to accommodate different models of high-level policies. First, domain experts translate high-level policies into specifications using ontology or formal vocabularies. This task has already become a part of the necessary responsibilities for IT department in every organization, because every organization needs to enforce their administrative and managerial policies. And the first step is to make machine "understand" these policies. Meanwhile, low- level enforcement mechanisms such as functions, services, protocols, and etc. have their specifications as well. Mapping mechanisms have been proposed to translate high-level policies to low-level mechanisms, such as top-down mappings and bottom-up mappings. Top-down mappings try to search corresponding features in low-level mechanisms for high-level policies. Bottom-up mappings present all the available mechanisms to the policy definer, and allow only enforceable features to be included in the policy. Bottom-up mappings usually need a good visualization to help a policy definer understand those low-level features and mechanisms. We integrate the advantages of both top-down and bottom-up styles to build an intermediate–level layer, and construct a comprehensive model-driven translation in the intermediate level to bridge the gap between two levels.

IV. ENFORCEMENT ARCHITECTURE

To complete the policy enforcement hierarchy, a model-driven enforcement architecture is proposed. Figure 3 illustrates the workflow of the entire enforcement architecture using UML. In this architecture several predefined models and procedures shown in figure 3 are used. We introduce the definitions of these models and procedures here first. Then we describe detailed steps of operations in the entire enforcement architecture.

Definition 1 Mathematical-or-logical model (MLM) is a collection of general operation rules used as standards by different domains, such as standard business rules or policies under contracts.

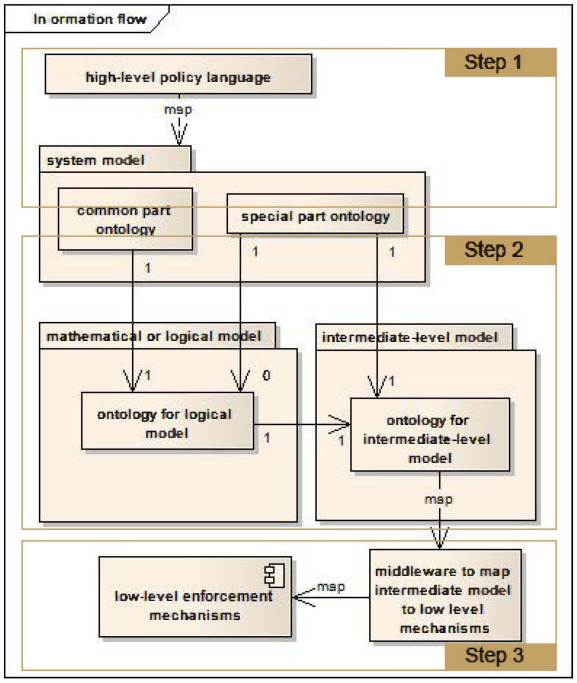

Figure 3. Information Flow of the Enforcement Architecture

Definition 2 System model (SM) is a formal representation of policy language specifications, which can be used to define management policies, security policies, privacy policies, or privacy configuration rules, such as EPAL, XACML, APPEL, and etc.

Definition 2.1 Common part ontology (SM_C) is a description of policy rules and their relationships that exist in common practices of most business applications or communities of business domains, which follow industrial standards or business contracts. It includes those elements having direct correspondence with the elements in a MLM.

- Assuming: SM_C is a set of common part ontology. MLM is a set of ontology for mathematical-or-logical model. SM_C and MLM have one-to-one mapping F1, which means every element sm_c of set SM_C has only one mapped image under F1, element lm of set MLM has one inversely mapped image under F1. This can be expressed as:

  F1 : SM_C ↔ MLM

Definition 2.2 Special part ontology (SM_S) is a description of policy rules and their relationships that uniquely exist in applications or requirements of certain domains or communities of domains, which may not be included in any industry standard. It includes those unique elements in the system model.

- Assuming: SM_S is a set of special part ontology. MLM is a set of ontology for mathematical-or-logical model. SM_S and MLM do not have one-to-one mapping under rule F2, which means any element sm_s of set SM_S does not have any mapped image under F2. This can be expressed as:

F2 : SM_S −−→ MLM

Procedure 1 Mapping procedure from system model to mathematical-or-logical model is defined as this: if an element or a relationship between elements in a system model has exact mapped one in mathematical-or-logical model, this element or relationship between elements will be included as a part of the common part ontology; if no mapped element exists, that element in the system model should be categorized as a part of the special part ontology.

- system model = {common part ontology} ∪ {special part ontology} ({common part ontology} ∩ {special part ontology} = Ø)

Definition 3 Intermediate-level model (ILM) is a formal representation to show the relationship between SM and MLM. It is in the format of an ontology. After the SM being mapped to the MLM, part of the elements or relationships can be mapped directly (in common part ontology), and other elements cannot (in special part ontology). Both these two parts are included in the ILM, which in turn will be used to construct translation between high-level policy languages and low-level enforcement mechanisms.

- Assuming: MLM is the ontology for mathematical-or-logical model. ILM is the ontology for intermediate-level model. MLM is added to ILM if and only if MLM has one-to-one correspondence with SM under rule F3. This can be expressed as:

  F3 : MLM → ILM iff SM_C ↔ MLM

- Assuming: SM_S is the special part ontology. ILM is the ontology for intermediate-level model. An element sm_s of SM_S is added to ILM under rules F4, which does not have one-to-one mapping between SM and MLM. This can be expressed as:

  F4 : SM_S → ILM

Procedure 2 Mapping procedure from SM and MLM to ILM is defined as this: the ontology of MLM is directly added to ILM together with its correspondence in the common part ontology of the SM, and the special part ontology of the SM is inserted into the ILM after the addition of MLM.

Definition 4 Low-level enforcement mechanism (LLM) includes those low-level functions, services, and configurations that are designed for the usage by local users and domain administrators for security protection, access control, privacy configuration, and other management purposes.

- Assuming: ILM is a set of ontologies for intermediate-level model. LLM represents a set of low-level enforcement mechanisms. ILM and LLM match with each other under rule F5. An element il of set ILM has only one image under F5; an element ll of set LLM has one inverse image under F5. This can be expressed as:

  F5 : ILM ↔ LLM

Procedure 3 Mapping procedure from ILM to LLM is defined as this: in the ILM, the actual elements in a policy language and relationships between elements with a match of the MLM are identified. These elements are searched in LLM to find matched functions or services, which can meet required properties and relationships in the MLM. Then these matched mechanisms are recorded in an OWL file. Unmatched elements and their required properties and relationships are also recorded in a separate part for further manual adjustments from administrators or system developers

V. SIMULATION

After defining all these necessary terms and procedures, we can go through the entire workflow of this three-layer enforcement architecture to simulate policy enforcement across domain boundaries. First of all, we have to obtain valid source information about the relevant selection of key characteristics and behaviors. The first key characteristic is high-level policy language. For different purposes of different domains, different policy languages are used to construct their system models. For example, in most cases, the privacy protection of social networking sites is regulated by access control policies. In healthcare environments, electronic medical records are protected under certain security policies. The second key characteristic is the mathematical or logical model. It can be obtained from common privacy configurations, business practices, or security rules. For example, the common privacy configurations of social networking sites include a common rule “only my friend can see my photo”; a common rule in healthcare environment “without obtaining the individual’s authorization, covered entities are permitted to utilize or disclose PHI (protected health information) to whom the PHI pertains, in case where the law requires such disclosure.” The third key characteristic is the low-level enforcement mechanisms. We need the enforcement mechanism information of involved domains for testing and verifying policy compatibility in policy enforcement simulation.

Meanwhile, like all simulations, certain simplifications and assumptions exist in our policy enforcement as well. First, we assume every policy can be expressed by a formal language, so that we can always get the system model. Second, we assume most elements in the mathematical or logical model, which reflect common practices or requirements, can be mapped to the system model, so that we can find the clear boundary between common part ontology and special part ontology. Third, we assume the intermediate level model should be able to fill the gap between high-level policy languages and low-level enforcement mechanisms by semantic mappings from both formal languages and machine languages to the intermediate level model. Based on these simplifications and assumptions, we can go through three stages in the enforcement architecture to simulate cross-domain policy enforcement.

The expected simulation results include how many policy rules are similar or identical to the partners’, how many local policy rules can be mapped to a partner domain’s enforcement mechanisms, and how many policy rules need manual coding to deploy. As a byproduct, partial enforcement code can be generated automatically. Before starting simulation, we need to do some preparation for key characteristics selection and certain simulation assumptions of the simulation. The mathematical or logical model should be formed and translated into a standard ontology language - OWL. As we know, ontology is a formal representation of a set of concepts within a domain and the relationships between these concepts. Ontology

can clearly represent all the elements and relationships in various models used in our architecture. We choose OWL because ontology is language independent and OWL is a family of knowledge representation languages for authoring ontology. Then the entire simulation can be performed following the steps described below. Figure 4 illustrates how the proposed enforcement architecture works for domain A and B. With the help of this simulation environment, an administrator or user can tell whether a policy set from local domain can be enforced in a partner domain's execution platform or vice versa.

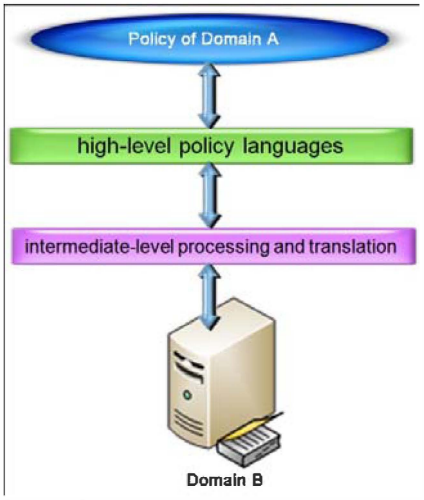

Figure 4. Simulation of Policy Enforcement across Domain Boundaries

The first step is to find a suitable high-level policy language and its system model to match the policy rule set of domain A if we intend to enforce domain A's policy as illustrated in figure 4. The usability of the entire enforcement architecture is affected by this system model because it directly defines what features are available and how can they be used. As mentioned in the definition of the mathematical or logical model, an appropriate policy language needs to be chosen for accommodating general operation rules or industrial standard rules used in multiple domains. This policy language together with its system model is one of the key characteristics in this simulation. After choosing the policy language, ontology is used to describe all the elements in rules and the relationships between rules. This ontology can be described by using OWL. In the Myspace example, we can include two elements and a relationship from Myspace domain (domain A) in this way, "birthday is the resource; friends are the subjects; reading action is the relationship between them."

The second step is to derive an intermediate-level model from the mathematical or logical model and the system model. In this step, elements and relationships in the system model ontology are merged with the mathematical or logical model for next step process. For example, the set of common elements used in privacy profile configurations for most social networking sites reflects general practices in this industry. The logical model accommodating these common elements is merged with the policy language (XACML)'s system model used for Myspace (domain A) privacy rule specification. In this step, we assume that both the system (policy) model and the logical model can be represented by ontology similar to that of typical social network elements. Then, elements in the intermediate-level model can be split naturally into a common part ontology and a special part ontology. The common part ontology includes the merged elements having direct correspondence between the system model and the logical model. The special part ontology includes those unique elements in the system model. We use a comprehensive mapping mechanism to translate the system model into an intermediate-level model, which utilizes ontology-based mapping and query-based mapping to find correlations between the system model and the logical model. The intermediate-level model is built from a tailored logical model (merging with system model elements) combined with certain extensions from unique system model elements.

The third step is to map the intermediate-level model to available low-level enforcement mechanisms from another domain (domain B) using query-based construction. Then this top-down mapping returns all the unsupported elements in the intermediate-level model back to the administrator (or user). So the user can amend this problem by modifying high-level policies or extend low-level mechanisms. There are two merits of this architecture. The first one is that users can choose their high–level policy languages to support the most usability they want; the second one is that the domain administrator can introduce a new core logical model when it is more appropriate for the enforcement task's target. In this step, one important requirement (assumption) is that low-level enforcement mechanisms should provide a clear specification of APIs for query-based construction of mapping between the intermediate-level model and low-level mechanisms. In the Myspace example, all the mapped APIs in the low-level mechanisms are matched with their corresponding elements in the intermediate-level model and recorded in an OWL file. Then this OWL file can be used to help generate enforcement code automatically for Facebook domain. Unmapped elements in the intermediate-level model are also recorded in the OWL file for further notification. Figure 4 illustrates the entire policy enforcement process.

After these three steps, this policy enforcement simulation environment can tell the system administrator (or user) whether a local policy set can be enforced in a partner domain. If not entirely, how many policy rules can be enforced in the partner domain's execution environment. As a byproduct, those mapped policies can be translated into enforcement code automatically. If too few policy rules are supported by the low-level mechanisms in the partner domain's execution environment, the administrator (or user) can decide to modify policy rules or work with partner domain's administrator for developing a plan to manually code unsupported policy rules. Thus, both the system administrator's and the developer's workload can be reduced dramatically.

VI. CASE STUDY

To illustrate the full capability of our proposed simulation framework, we discuss a real world application in this section. Our objective is to provide a simulation environment for evaluating the possibility of cross-domain policy enforcement. Policy modeling and partial code generation are by-products that can be used for real enforcement in future development and deployment. Through the case, we will show how the policy model and mathematical or logical model are formed; how

partial code is automatically generated; how we can apply this simulation environment in different applications using its different aspects? In this case study, the entire simulation architecture is applied in a healthcare environment. In this environment, system administrators need to define policies following HIPAA and other regulations for all electronic medical records and other digitized information; doctors and medical specialists can define security policies for medical records; and patients can define their own security policies and access control to authorize utilization or disclosure of their own information.

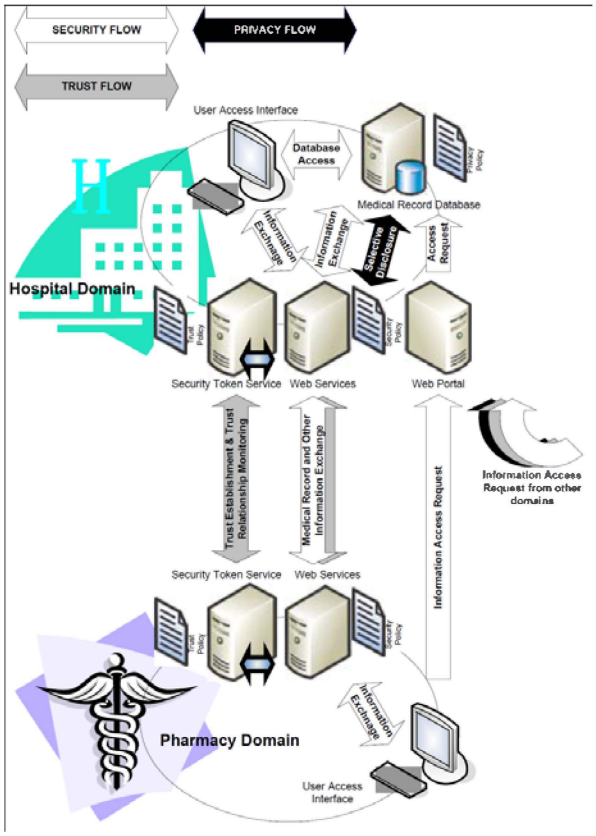

Figure 5. Federated Information and Control Flows Between Hospital and Pharmacy Domains

As illustrated in figure 5, system interactions between the hospital domain and the pharmacy domain, and different components within one domain are through web services. Security policies for security flows are described in WS-SecurityPolicy and WS-Security formats, which are used for security protections of medical records and access control of other patient information. Trust policies for trust flows are described in WS-Trust format for cross-domain federation activities. Privacy policies for privacy flows are described in WS-Policy format for privacy protections within and across domains. These policies need to be enforced in local domain as well as in domains involved in interactions. Our simulation environment can help system administrators decide the possibility of policy enforcement across domain boundaries before real policy deployment. As a byproduct, partial enforcement code can be generated automatically. For example, the system administrator at the hospital domain defines a set of security policies to regulate access of different types of healthcare information through web services. Then, to facilitate federation activities across domains, the hospital domain and the pharmacy domain both need to define trust establishment policies and token exchange policies for negotiating trust and validating trust relationships also through web services. Besides, to protect their own privacy, patients also can define a set of privacy policies to selectively disclose their healthcare information from the hospital domain to the pharmacy domain if it is necessary. White, grey, and black arrows represent these three types of information and control flows respectively. Our simulation environment can help analyze the possibility as well as an estimated workload to enforce these different types of policies over domain boundaries.

A. Logical model of healthcare systems

Privacy policy enforcement is especially important in healthcare systems, since HIPAA includes a clear declaration of patient privacy requirements. Protected Health Information (PHI) under HIPAA is individually identifiable health information. Identifiable information not only refers to data that is explicitly linked to a particular individual but also includes health information with data items that can reasonably be expected to allow individual identification. Under HIPAA "safe harbor" standard, information is considered de-indentified if all of the above have been removed, and there is no reasonable basis to believe that the remaining information could be used to identify a person.

There are also other HIPAA rules for Security, Identifier, and Transaction and Code Set, such as "without obtaining the individual's authorization, covered entities are permitted to utilize or disclose PHI to the individual to whom the PHI pertains, for purposes of TPO (healthcare operations), to another covered entity for the healthcare operations of the entity receiving the information, with valid authorization, if the covered entity has received the individual's oral agreement for the use of the PHI, and in instances where the law requires such disclosure." HIPAA also includes clear definitions for security and trust policies in healthcare environments, such as person or entry authentication, workforce security, transmission security, security management policies, and etc. All of these policies can be translated into formal mathematical or logical model for enforcement.

B. System model of healthcare systems

In the healthcare environment, the system model for three types of policies needs to be established for simulation. For privacy policies, we use XACML policy language to represent and build system model with a similar procedure described in section 6.1. For security policies, due to the fact that healthcare systems distribute their privileges to different roles, there are different ways to support confidentiality, integrity, authentication, and other security services. So we use the formal WS-Security and WS-SecurityPolicy specification to build the system model for these requirements. For trust policies, WS-Trust specification formally describes the way trust can be established and maintained in the web services environment. So it is used to construct the trust-related part in the system model.

C. Implementation

Following the simulation architecture, we generate the system model from specifications of XACML, WS-Security,

WS-SecurityPolicy and WS-Trust into an OWL-formatted ontology as the first step. Then the system model maps to the logical model and returns a common part ontology and a special part ontology. Both of these two parts are included in the intermediate-level model. Finally, we use a query-based construction to map the intermediate-level model to low-level enforcement mechanisms. The possibility of cross-domain enforcement is then determined by the domain administrator considering how many policy rules don't have supporting low-level mechanisms, how much code is automatically generated by the simulation environment, and how much code still needs manual development.

VII. DISCUSSION

The core of this simulation environment is the proposed new policy enforcement architecture, which can evaluate potential cross-domain policy deployment through model-driven mapping and translation. The critical part of the policy enforcement architecture is the intermediate-level modeling and translation, which transforms high-level policies into formal models and maps these formal models to low-level enforcement mechanisms. This enforcement architecture can not only simulate cross-domain policy enforcement but also have the potential to be used in real policy development and deployment.

The major contribution of this policy enforcement architecture is in workflow innovation. Traditionally, policy development and deployment need three steps – policy rule definition and formation (by administrative personnel), policy rule translation (by technical staff), and enforcement code development (by programmers). But the gap between step two and step three needs substantial knowledge and experience for programmers. Our enforcement architecture tries to absorb the knowledge from the technical staff to build a policy model for each policy language and automate the tedious translation process (code development) from policy language to executable code using semantic mapping and query-based mapping. Compared with traditional approaches, our enforcement architecture connects high-level policy languages to low-level enforcement mechanisms by using an automatic model-driven process. Meanwhile, enforcement code previously requiring manual development can be generated automatically if proper APIS or formal descriptions for low-level mechanisms are available. For those unmapped elements in the intermediate-level model, our enforcement architecture can also estimate future manual coding effort. But, on the other side, we require an application of our enforcement architecture should provide a formal model for each type of policy and each policy language, or support formal modeling of existing policy languages. We also assume a proper API exists for low-level mechanisms if code generation is desirable.

In the implementation of the case study, policy modeling, mapping and transformation are transparent to user. The graphic user interface can help user monitor the correctness of mapping and transformation. The effectiveness of policy enforcement simulation is predicated on the correct construction of mapping rules. Use of semantic ontology language to represent models used in different steps such as Web Ontology Language (OWL) can guarantee correct processing in the entire simulation. As long as policies can be correctly represented in ontology [2], whether security policies, trust policies, authorization policies or privacy rules that need modeling and processing does not matter. The usability of this simulation environment can be further improved by providing more user monitoring interfaces [4,5].

A comparison with other enforcement architectures can help illustrate merits as well as identify this architecture's potential applications in real (future) development and deployment of policy enforcement systems. We list two representative enforcement architectures below for comparisons.

A. An enforcement mechanism for run-time security policies - In this mechanism [8], policies can be enforced by monitoring and modifying programs at run time, such as Edit Automata [2]. In Edit automata, program monitors are abstract machines that examine the sequence of application program actions and transform the sequence when it deviates from a specified policy. Security properties are enforced in this mechanism by a monitor program that runs in parallel with a target application program. Whenever the target program wishes to execute a security relevant operation, the monitor first checks its policy to determine whether or not that operation is allowed. If the target program's execution sequence is not in the property, the monitor transforms it into a sequence that obeys the property. This mechanism has two major considerations. This first one is that the final output of a monitored system must obey the policy. Consequently, bad programs that would otherwise violate the policy must have their executions modified by the enforcement mechanism. The second one is transparency, which means whenever the un-trusted program obeys the policy in question, a run-time enforcement mechanism should preserve the semantics of the un-trusted program. But it still requires expert level knowledge of security properties and policies so that an interpreter is still needed to use this mechanism.

B. Antigone system - In Antigone system [9], there are three levels of policies are defined for communication systems: application-level policy, enterprise-level policy, and session policy. The Antigone framework fills the gap between policy representation and enforcement by implementing and integrating the diverse security services needed by policies. Policies are enforced by run-time composition, configuration, and regulation of security services. Antigone does not implement policy-enforcing software, but provides APIs and an associated framework for its definition and use. A central element of the Antigone enforcement architecture is a set of mechanisms that provide the basic services needed for secure groups. Policies are implemented by the composition and configuration of these mechanisms. Thus, Antigone does not dictate the available security policies to an application, but provides low-level mechanisms for implementing them. The centralized control mechanism needed for all enforcement activities in Antigone system can neither be distributed nor be applied across domain boundaries. The semantic gap between policy languages and enforcement mechanisms also exists in Antigone system.

Both systems described in A and B (system (2) and (3) thereafter in this section) have their own advantages and disadvantages. But to understand the merits of new enforcement architecture (system (1) thereafter in this section)

used in our simulation environment, we need to compare this new architecture with system (2) and (3) form different aspects listed in Table 1. From the aspect of architectural hierarchy, system (1) and (3) use a three-level hierarchy, which helps to construct a bridge over the gap between the top-level and bottom-level. System (1) is also policy independent. For storage, system (1) uses OWL representation file to store its policy models. System (2) stores and monitors its configuration files. System (3) maintains its session-specific policy instances. For core operations, system (1) is based on a web environment for cross-domain enforcement, and system (2) enforces only system properties, which the core mission of system (3) is to regulate sessions and subsequent system provisions. For usability, system (1) can be performed automatically or manually; system (2) has to have supports from operating systems; system (3) has the limitation that its users have to develop software using its Antigone framework through its APIs. The time complexity of system (1) is O(n2), while for both system (2) and (3), the time complexity becomes a NP-Complete problem. For dynamicity or flexibility, *system (1) and (2) support dynamic policy enforcement.*

	(1) Our enforcement architecture	(2) Enforcement mechanism for run-time security policies	(3) Antigone system
Hierarchy	Three levels	Two levels	Three levels
Policy Independence	yes	no	no
Storage	OWL representation file of policy models	configuration file monitors	session-specific policy instance
Operation	Our architecture provides a web-based environment for monitoring cross-domain enforcement	monitors are used to enforce system properties	A session-specific policy instance is created by an initiator through the reconciliation algorithm. The instance is subsequently used to regulate the session and subsequent provisions
Usability	Automatically mapping high-level language to low-level enforcement architecture + Manually assistance mapping correction	The monitor must has support of operating system	Users have to develop software base on Antigone framework by using the APIs.
Time complexity	$O(n^2)$	NP-Complete	NP-Complete
Dynamic policy enforceable	yes	yes	no

Table 1. Comparison of Different Architectures

VIII. CONCLUSION

Policy-based management provides a flexible way for security management, privacy protection and adaptive access control, when multiple domains cooperate or collaborate to finish a common goal. It requires system administrators to consider the possibility of integrating or interconnecting two or more domains when these domains have different policy definitions and different policy enforcement mechanisms, and estimate the workload for this cross-domain policy enforcement effort if it is necessary. This paper introduces a simulation environment to help to do so to evaluate the possibility before software development or system rebuild. The central part of this simulation environment is a new enforcement architecture to provide an intermediate-level component for mapping processing and configuration recording. Each pair of domains can establish one intermediate-level component for cross-domain enforcement. Once this intermediate-level component is created, it can be re-mapped and manually modified at any time. This intermediate-level component can also be extended to more than two domains. Then the entire request from partner domains will be processed and mapped through this enforcement architecture to decide the possibility for cross-domain enforcement. Code can also be automatically generated based on these mapping results. Following the study cases, the advantages of this enhanced new enforcement architecture are confirmed and can be summarized into three merits: Administrators and users can choose their high-level policy languages with the most expressive capability; domain administrators can change the core mathematical or logical model when it is more appropriate for system controls or low-level enforcement mechanisms; the translation and mapping in the intermediate-level is flexible.

REFERENCES

[1] V. Crescini, Y. Zhang, W. Wang, "Web server authorization with the policyupdater access control system," Proc. 2004 IADIS WWW/Internet Conference, vol. 2, 2004, pp. 945–948.

[2] Vladimir Kolovski, Bijan Parsia, Yarden Katz, and James Hendler, "Representing Web Service Policies in OWL-DL," Y. Gil et al. (Eds.): ISWC 2005, LNCS 3729, 2005, pp. 461–475.

[3] OASIS, "eXtensible Access Control Markup Language (XACML) Version 2.0," Feb. 2005, http://docs.oasis-open.org/xacml/2.0/access_ control -xacml-2.0-core-spec-os.pdf

[4] Uszok, A., Bradshaw, Lott, J. Breedy, M., Bunch, L.,Feltovich, P., Johnson, M. and Jung, H., "New Developments in Ontology-Based Policy Management: Increasing the Practicality and Comprehensiveness of KAoS," IEEE Workshop on Policy 2008, IEEE Press.

[5] Uszok, A., Bradshaw, J., Jeffers, R., Johnson, M., Tate A., Dalton, J., Aitken, S., "KAoS Policy Management for Semantic Web Services," In IEEE Intelligent Systems, Vol. 19, No. 4, July/August 2004, pp. 32-41.

[6] Ralph Gross, Alessandro Acquisti, "Information Revelation and Privacy in Online Social Networks," ACM Workshop on Privacy in the Electronic Society (WPES), 2005.

[7] Catherine Dwyer, Starr Roxanne Hiltz, "Trust and privacy concern within social networking sites:A comparison of Facebook and MySpace," Proceedings of the Thirteenth Americas Conference on Information Systems, Keystone, Colorado August 09 - 12 2007.

[8] Jay Ligatti, Lujo Bauer, David Walker, "Edit automata: enforcement mechanisms for run-time security policies ", International Journal of Information Security, Volume 4, Numbers 1-2, February 2005, pp. 2-16.

[9] Patrick McDaniel, Atul Prakash, "Enforcing provisioning and authorization policy in the Antigone system", Journal of Computer Security, Volume 14, Number 6, 2006, pp. 483-511.