Dynamic Cross Domain Information Sharing- A Concept Paper on Flexible Adaptive Policy Management

Michael Atighetchi, Jonathan Webb, Partha Pal, Joseph Loyall
Raytheon BBN Technologies
10 Moulton St, Cambridge, MA, USA
001-617-873-1679
{matighet,jwebb,ppal, jloyall}@bbn.com

Azer Bestavros
Department of Computer Science
Boston University
Boston MA, USA
best@cs.bu.edu

Michael J. Mayhew
Cross Domain
Innovation and Science Group
Air Force Research Laboratory
Rome NY, USA
michael.mayhew@rf.al.mil

ABSTRACT
Information exchange across domains is essential for today's asymmetric warfare environment to make mission-critical information available to war fighters, no matter where it exists and when it becomes available. Dissemination of new information needs to carefully balance the need-to-know by consumers with the responsibility-to-share by providers. The right amount of sharing, governed by policies defining what information can cross domain boundaries, when, and under what circumstances, is highly context-dependent and dynamic. Dynamic management of those policies is a key challenge. This paper describes the design of concepts and services to support dynamic lifecycle management and deconfliction of policies governing cross domain information flows. We describe how the design provides scalable, on-the-fly reconfiguration of both local and cross domain security policies while confining sensitive policy information to their respective local domains.

Categories and Subject Descriptors
D.4.6 Security and Protection, K.6.5 Security and Protection

General Terms
Algorithms, Management, Design, Security

Keywords
Dynamic Policy Management, Cross Domain Information Sharing

1. INTRODUCTION
Unauthorized disclosure, sharing, and modification of sensitive information are critical issues in network-centric systems as the value, volume and variety of information flowing through the systems increase. The need to share such information is also rising in order to support emerging applications requiring rapid set up and dynamic reconfiguration of sharing requirements.

Controlling networked information flow solely by static enforcement of security policies like the "no read-up, no write-down" rule of the classical Bell-La Padula [1] model is becoming untenable because of the increasing need to seamlessly handle dynamically unfolding events. Such events include amendment to existing policies because of organizational changes or availability of new services, coalition formation, changes in the priority and urgency of specific classes of information or information exchanges, e.g., due to geo-political events, or elevation of risk/threat profile of a certain region or subjects, e.g., based on intelligence reports. In addition to actual insertion or deletion of policy rules, such changes often require changing attributes of subjects and resolution of various identities. Any modification to the policy mechanism must be integrity preserving; in particular, there must not be any unauthorized disclosure (leakage) or denial of permissible information access (blockage). That is why (re-) accreditation of policy (changes) is so critical. Accreditation is a manual process that takes months and in many situations such delays between the change trigger and deployment of the modified policy are not acceptable.

What is needed is a capability that enables switching between policies within minutes without introducing new risks or vulnerabilities through a system that provides dynamic authoring, selection, and deployment of security related policies of cross domain solutions (CDSs). Technologies for providing such functionality within domains are readily available in today's enterprise environments. Examples include the role-based access control services of the Net-Centric Enterprise Services (NCES) [2] and network management platforms based on SNMP that monitor and control configurations of IT assets, such as hosts and routers. However, COTS policy management solutions cannot be directly applied to the cross domain context for a number of reasons.

This paper first outlines a solution that enables rapid and safe deployment of appropriately updated policies responding to policy change drivers in a cross-domain environment, and then presents the fundamental concepts underlying the solution. The main vision is to decouple the authoring of accredited policies from deployment. Construction of new or modified policies for various contingencies, e.g., coalitions or joint operations, anticipated troop or command center migration, and changes in threat levels, and their subsequent accreditation are done off line as planning exercises. The resulting policies are appropriately tagged and indexed in a secure and persistent store. Automated policy analysis is a key enabler for constructing consistent policies to begin with, which reduces the time spent in the authoring cycle. When a situation arises, and conversely when a situation normalizes, an automated process assists operators to quickly identify appropriate policies based on the requirements at hand and the tags of already accredited policies from the persistent store. Once identified, selected policies are pushed out to appropriate CDSs reliably so that

SafeConfig'10, October 4, 2010, Chicago, Illinois, USA.

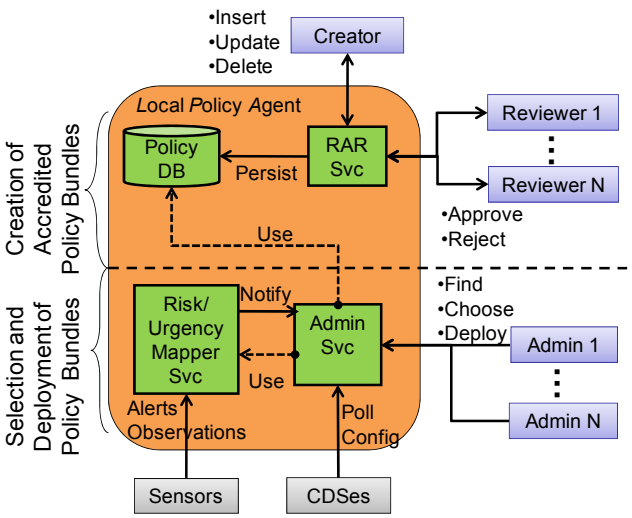

Figure 1. A Functional View of the Local Policy Agent (LPA)

the new policies become effective quickly by means of domain-specific enforcement mechanisms.

2. DYNAMIC POLICY MANAGEMENT OF CDSs

We have identified the following key workflows required to implement flexible and adaptive policy management for cross domain environments:

- Authoring & Persistence: Enable construction and persistence of policy sets ranked by urgency/risk profiles through reliable accreditation review (RAR) workflows.
- Context-based Selection: Provide an easy to use interface which allows administrators to choose an appropriate policy given a certain risk/urgency profile. In addition to manual selection via browsing, the system also needs to automatically evaluate changes in current status derived from sensors inputs and suggest policy reconfiguration actions to the administrator.
- Deployment: Provide generation and deployment of Data Flow Configuration File (DFCF) [3] descriptions with transactional deployment semantics to recover from partial failures during deployment of policy updates to enforcements devices.

2.1 Challenge Problems

Any system implementing the three workflows in the cross-domain context must address the following difficult challenges:

Policy Representation: How to transcribe existing policies into forms amenable to automated analysis? Resulting policy representations need to be expressive enough to capture the complexities of the security policies and concrete enough to enable systematic analysis. Representations will also need to support both hierarchical and non-hierarchical domain relationships and allow for self-referential policies, i.e., policies that describe restrictions on themselves.

Secure Persistence and Tagging: How to enable trustworthy interactions with the policy management system through strategic use of protection and auditing mechanisms? This includes secure storage of pre-approved policies in a trustworthy way for fast retrieval, authentication of humans, and protection of message exchanges through standard protocols, e.g., TLS.

Control Locus: How to support both dedicated and hierarchical management layouts? Multi-domain management solutions need to get appropriate requests into and appropriate information out of domains with restrictive data-sharing policies that are common in complex multi-level security and coalition environments.

Scalability: How to minimize the proliferation of point-to-point CDS deployments while scaling to a large number of supported domains, so that the resulting integration remains manageable? Furthermore, the solution must manage the state explosion typically encountered in analyzing a large number of policies of real-world complexity.

Interoperability: How to support a diverse set of existing CDSs, including security guards and gateways, by utilizing existing and evolving configuration standards, e.g., DFCF?

Automating Policy Lifecycle with Reliable Accreditation Review: How to minimize human involvement? This relates to the process of dynamically modifying cross-domain policies to respond to change drivers while retaining the capability of RAR at critical workflow points such as authoring, accreditation of a policy bundle prior to persistent storage, and selecting appropriate policy bundles. Applicable authorities may require multiple signoffs for critical operations in a policy lifecycle. Automation should assist the operators wherever possible, including tool support for policy analysis, automatic capture of audit trails, and automatically ensuring the transaction semantics for configuring CDS enforcement platforms.

Certification and Accreditation: How to avoid the need for certifying and accrediting the entire policy management system? A major issue in streamlining the change management process is to validate that the dynamic management capability itself does not introduce new leakage of information from high to low domains or create new attack vectors from low against high domains.

2.2 Concepts Solving the Challenge Problems

In the Lifecycle Management, Deconfliction, and Automatic Deployment Services for Cross Domain Security Policies (LDADS for XSP) project, we are creating an enterprise capability to dynamically and securely adapt CDS policy to changing operational risks and requirements. This paper describes the LDADS architectural concepts and designs. Creation of a proof-of-concept prototype is currently in progress.

To support the three functional flows, we introduce the notion of a ***Local Policy Agent*** (LPA) for providing dynamic policy management within a domain. One can think of the LPA as a domain's centralized service for managing cross domain policies and configurations. Figure 1 displays a functional view of the LPA. Functionality is split into the parts of 1) creation of accredited policy bundles (top half) which can take significant time (in the order of months) and 2) selection and deployment of those policy bundles, which needs to be fast (in the order of minutes). As shown in the figure, the policy creators and reviewers interact with the LPA through a ***RAR service*** that supports the standard lifecycle operations of create, update, and delete, as well as operations to approve or reject changes. To support these operations, the RAR component is designed in a modular way through a combination of new and existing services, including voting services, RSS notification services, and provenance services. It is our intention to utilize existing capabilities such as PMAF [4] or

DDT/WES [5] and define additional requirements in case needed functionality doesn't currently exist. The RAR service needs to allow interactions with creators and reviewers for authoring of policies. The RAR component interacts with a ***Policy Database (DB)*** to persist policies throughout their life cycle.

The LPA supports dynamic selection and deployment of pre-approved policies through a sensor-actuator control loop with human involvement. Sensor inputs, such as DoD threat levels or significant attacks flagged by local intrusion detection systems, enter the LPA and get dispatched to a ***Risk/Urgency Mapper service*** which translates the specific sensor information into a normalized risk urgency matrix. The Mapper then notifies administrators through visible events in the ***Administration service*** and alert messages sent via RSS or email. An administrator can then manually browse the database of pre-approved policies and select a policy that satisfies the current risk/urgency status. Alternatively, the Administration service provides functionality to automatically assist the Administrator in finding appropriate policies and learns to refine relevant contexts for selecting policies over time to speed up the selection process. Once a proper policy is selected, the administrator can assemble DFCF configuration descriptions and start deployment.

Policy modification requires a ***transaction model*** coupling changes to the rule base with the currently enforced version of the policy. For example, the introduction of a new rule may generate a conflict with existing policies, which needs to be resolved through deconfliction (as described in the next paragraph). Parallel execution or uncontrolled interruption of the update process may leave the currently deployed policy in an inconsistent state. We intend to address such synchronization problems by developing a transaction model for policy updates and deployments, accommodating locking and rollback of updates to avoid update inconsistencies. We further intend to allow for preemption of service invocations during active policy updates to avoid inconsistencies in policy enforcement. Support for transactions allows the LDADS system to handle error conditions as soon as they occur as opposed to falling back on manual debugging of inconsistent deployments after the fact.

LDADS allows a variety of deployment scenarios to support scalability with a variety of scale regimes. Figure 2 displays a small scale deployment scenario with a few domains and CDSs. In this case, the LPA in Domain 1 can be configured to directly control the CDSs of its local domain without further coordination with LPAs in other domains, such as the LPA in Domain 2. However, to get consistent overall policies, this approach requires administrators in both domains to synchronize their actions out-of-band by either selecting one of the LPAs as the master or manually synchronizing respective partial updates in cases in which the domain's outbound policy is restricted and not-sharable across the domain boundary. This deployment scenario becomes quickly unmanageable with three or more domains and CDSs.

To support more scalable deployments, LDADS is built on top of a hierarchical policy management architecture based on architectural patterns developed under the XDDS effort for performing cross-domain service discovery. In that architecture, the LPA forms the locus of a domain's policy management functions, as displayed in Figure 3. Specifically, dynamic policy changes in Domain 1 are handled by the local LPA in Domain 1, which transparently hooks into the local domain and provides functions to 1) analyze local cross-domain policies and make analysis results accessible to other domains but only through appropriate CDSs, and 2) deploy policy changes into edge gateways and security guards. The LPA uses purpose-built policy update protocols that accommodate requirements on message exchanges in cross-domain environments.

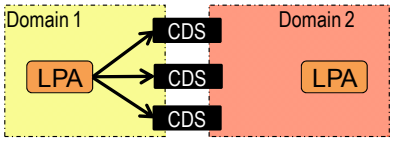

Figure 2. Local Policy Agents Dynamically Reconfigure CDSs

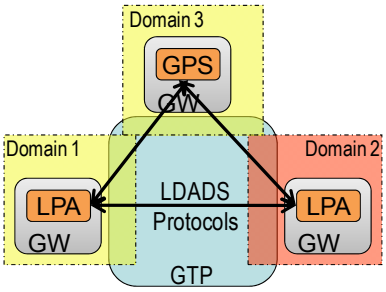

Figure 3. Hierarchical Architecture

The interplay between LPAs in multiple domains is coordinated by a ***Global Policy Service*** (GPS), which controls and arbitrates communication between LPA instances and also provides anonymization and 3rd-party authentication for those domains that need them. The GPS resides in a domain of its own with a classification level that can accommodate classification levels of participating domains. Examples in this document assume a hierarchical domain structure and the GPS domain is assumed to be the highest among the participating domains—this has the advantage that the GPS can store information about all other domains without violating any access control and data-sharing policies. Both LPAs and the GPS are implemented as services and hosted in gateway components (GW) in respective domains. Our plan is to reuse existing gateways, such as the ICASE [6] IPS Tomcat containers or the Collaboration Gateway [7].

We keep the LDADS architecture guard-agnostic by encapsulating existing guard functionality into an abstract concept of the ***Guard Technology Platform (GTP)*** with well-defined interfaces. The function of the GTP is essentially restrictive. Like a firewall, the objective is to block or prevent traffic that violates its policies. There are a number of requirements on the underlying GTP layer that are necessary for supporting LDADS. First, the domains that participate in cross-domain collaboration need to have at least one GTP between any pair of domains that communicate so that information flows between LDADS nodes are adequately protected. Furthermore, the new kind of information (e.g., DFCF policies, compositional analysis results, etc.) that flows across domains needs to be appropriately labeled and signed.

Changes in policies will inevitably lead to policy inconsistencies that need to be mitigated through a harmonization process. Our approach applies research performed at Boston University on ***compositional analysis*** [8][9][10] to the authoring of policies through the RAR service to detect policy inconsistencies and identify candidate rules to change. Composing individual policy analyses rather than analyzing a large composite policy, allows us to improve scalability both in terms of the number of policies and the number of rules per policy. Compositional analysis also enables local processing on sensitive policies that are not sharable across domain boundaries. Using *type checking*, our system can calculate whether a particular composition of policies violates any constraints. *Type inference* allows us to develop more concise rules for policy composition across domains and *type debugging* allows us to identify required changes to conflicting rules more

precisely than is typically possible with conventional theorem proving. This allows the creators to iteratively debug policies before sending them off to reviewers, therefore minimizing cases in which policy changes would have been rejected by reviewers. Note that while we expect to implement a system-wide harmonization processes as part of the GPS, the same tool support is also available through LPAs in deployment scenarios with a small number of domains (such as shown in Figure 2).

An approach that uses raw predicate calculus based representations can easily get out of control as it scales with the number of predicates and the complexity of the relationships between terms in the expressions. This *complexity* is expected to be very large for models of security classification guides even with only a few domains and relevant policy rules. Compositional analysis addresses this problem by introducing types and type relations as a way to control state explosion. This approach scales with the number of types and rules and the complexity of their interactions and this scaling can be controlled by careful design of the type system.

This approach scales with the number of types and rules, which is much smaller than the number of predicates. As displayed in Figure 4, each LPA will host a dedicated type checker that analyzes policies for local consistency and shares cross-domain **results via a small set** of types. These results are shared with a GPS in a dedicated domain, leading to a scalable hierarchy of LDAs and a GPS. Alternatively, results may also be shared directly between peer LPAs in the absence of a GPS.

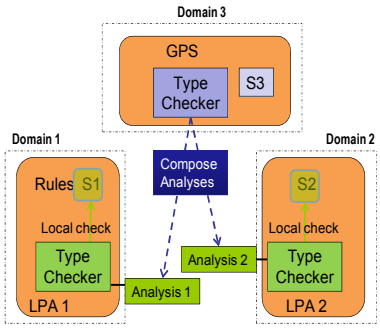

Figure 4. Compositional Analysis

Machine executable ***policy representations*** need to be expressive enough to capture the complexities of the security policies at hand and concrete enough to enable systematic analysis. A promising approach is to interface to policies expressed in open standards (e.g., XACML) through a human-assisted process that ingests policies and translates those policies into an internal representation based on type hierarchies and typing rules. The policy is divided into generic and domain specific rules. Rules that govern intra-domain interactions are separated from rules that restrict cross domain interactions. Domain relationships can be hierarchical or non-hierarchical, and policies can describe restrictions on themselves.

Addressing ***interoperability*** is key, since any solution needs to interface with existing policy frameworks and scale as the number of domains and policies increases. One approach we have used successfully in the past is to encapsulate existing guards via GTP interfaces to provide a guard-agnostic solution.

Certification and Accreditation of CDSs is expensive and time consuming, and solutions that do not account for the specifics of cross-domain environments will face significant barriers during certification and accreditation. To address these issues, we are designing LDADS in a way that decouples existing policy management technologies found locally in a domain from the messages that cross the domain boundary. This keeps unnecessary complexity at the edge, while allowing the LPA/GPS components to exchange a smaller set of core messages within a narrowly defined message format (e.g., it will not support full-blown XML), derived from open standards. The resulting *generalized communication models* are part of the LDADS solution and, once certified, can be extended to work with a wide variety of policy frameworks through adapters.

3. DESIGN CONSIDERATIONS

Figure 5 displays the main core technology pieces as green boxes and indicates their relationships to each other and to the three critical workflows of authoring & persistence, context-based selection, and deployment of policies.

3.1 Transactions

Part of ensuring the overall reliability of the operation of the various workflows is to define a collection of transactional invariants and implement appropriate mechanisms for enforcing those invariants. For example, policy bundles will contain a variety of components and installation of the bundle will require all of those components be installed correctly. If a component fails, the operation must be retried or the CDS configuration must be rolled back to a previous state. Also, the transaction definition associated with the policy change must have access to control mechanisms in the CDS to interrupt or disable data transmission to ensure a transfer is not performed in an inconsistent policy state during installation of the new configuration. We plan to design the transaction management base technology layer to identify the necessary transactions in the lifecycle processes and develop appropriate enforcement mechanisms.

Transaction management tools are available in a variety of environments, J2EE containers like Tomcat and JBoss and transactionally aware database engines like Oracle and PostgreSQL being particular examples. Our goal is not to develop the transaction processor but to exploit existing capabilities and to deploy the relevant transactional applications for policy management within these environments. For example, in the XDDS work, the baseline capabilities are being developed with Tomcat and this platform would provide suitable transaction management capabilities for the workflows in policy lifecycle management.

- A major element for the design is identifying the transaction boundaries for various operations in the different lifecycle processes. Associated with each transaction are the necessary invariants describing the integrity conditions for the transaction. Some obvious transactions and their associated invariants are listed below.
- Installation of a new policy bundle or modification of an existing policy bundle is an obvious transaction. Part of the invariant is a completeness check for the content of the bundle. For example, if DFCF is used to represent a policy bundle, the supplied bundle must pass a schema check for the representation of the data. Also, credentials must be provided corresponding to the identities of the required reviewers for the policy bundle as it passed through the review process.
- Association of an ingested policy bundle with a particular risk or urgency profile will be performed in a transaction. The invariant for the transaction would include a requirement for only having a single default policy bundle and only a single policy bundle for each distinct risk and urgency value.
- Many actions will have auditing requirements. A common component of the invariants for the transactions surrounding

these actions will be the generation of the audit record and its persistent storage (show in Figure 5).

3.2 Auditing

Audit trails for various lifecycle events are an important element for any system of this type and will play a role in certification and accreditation processes as well as ongoing operation. Audit data represents a particular component in the data management process and will also have integrity requirements analogous to the policy bundle data integrity discussed previously. Auditing activities will also participate in the definition of transactions so audit data artifacts have appropriate accuracy constraints relative to actual changes in system state. The goal of the design of the auditing base technology layer is to identify auditable actions and store the relevant audit entries in a verifiable and secure manner.

A necessary feature of any auditing capability deployed in this environment is to provide integrity and continuity checks. These must be balanced against storage costs. Integrity checks on the logs will also help to identify tampering or errors in local processing.

It is desirable to exploit any existing services providing audit capabilities where possible. Most database engines provide extensively configurable auditing processes although log integrity is typically only as good as that provided by the file system of the host operating system. For LDADS, we are integrating existing auditing services or specify requirements for construction of appropriate services to use.

State changes having an impact on the policy definitions or CDS configuration will require corresponding audit entries with reference to the altered policy bundle, some digest representing its contents, an authorized identity making the change, and utility information like timestamp or sequence numbers. Administrative services need to be defined to support audit log maintenance and retrieval as well as alert mechanisms to identify system errors or other conditions requiring the attention of support staff.

3.3 Process and Data Integrity

The integrity of various process and data elements is required at selected points in the workflows. For example, human review is required during the policy ingestion phase of the authoring workflow for the user to validate for the system the accuracy of the supplied policy bundle relative to the user's intentions. This validation must include some representation of the identity of one or more reviewers of the supplied policy bundle before the bundle can be considered accepted for use by the system. Similarly, the process element responsible for installing the designated policy bundle in the deployment workflow must be able to validate the integrity of the workflow based on relevant reviewer authenticity, integrity of the policy bundle due to damage or alteration, and validation of the intent of the administrator requesting the change. The goal of the design of the integrity base technology layer is to provide an end-to-end validation chain for policy bundles from ingestion to deployment, as shown by dotted lines in Figure 5.

At least some of the technology elements supporting process and data integrity are expected to come from existing systems and services. For example, the DDT/WES project provides mechanisms for RAR in a cross domain context. Identity management, provenance services, and standardized encryption and digesting technologies are all likely to play a role and these existing elements need to be identified and exploited.

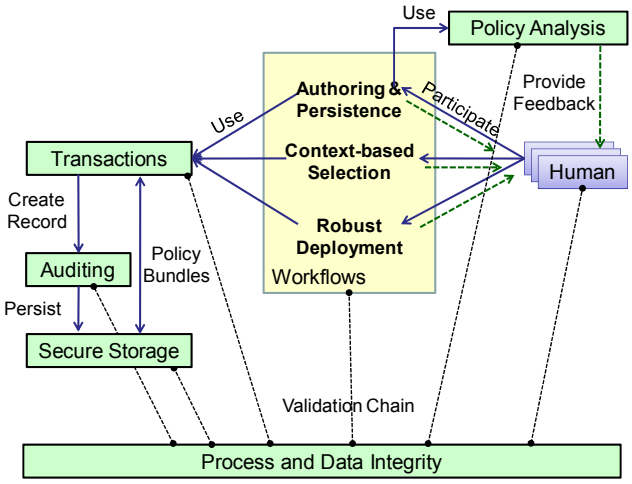

Figure 5. Core LDADS Technology Pieces and Their Relationship

A major element of the design problem for integrity is identifying the checkpoints in the various workflows where an integrity check is required and the necessary elements in the integrity package used to determine the integrity of the relevant action or policy bundle. Some obvious checkpoints are listed below.

- At policy ingestion time, the policy bundle itself must have an identification with some appropriate set of reviewers prior to being made available for use within the system.
- When policies are updated or modified, the existing policy bundle, updated policy bundle, and associated reviewers must be identified and validated.
- When risk or urgency profiles for a policy bundle are changed, the relevant policy bundles and mappings must be associated with a reviewer and the relevant policy bundles.
- When a policy bundle is selected for deployment, checks are required to ensure the accuracy of the mapping and the intent of the relevant administrator for the configuration change.
- When a policy bundle is deployed, the integrity of the provided bundle must be verified and supplied identities for associated reviewers and administrators must pass currency checks.

3.4 Policy Analysis

Our concept for policy analysis entails encoding of relevant policies derived from security classification guides (SCGs) as a collection of types and rules that are used by a proof engine for automated analysis. Types in a type system can be thought of as representing collections of statements in a predicate calculus. Operations on those types define compositional properties of the associated types in a manner consistent with the underlying expressions in the predicate calculus. The goal of the design of a type system is to be able to answer particular questions over this predicate calculus such as, in this case, whether or not a particular cross domain information flow is allowed. The validation of the type system is, of necessity, mechanical and follows the rules of the underlying predicate calculus resulting in a sound reasoning system. The design problem is to construct a type system with sufficient expressiveness to capture the questions of interest but with desirable scaling properties. The art of the design of a particular type system is to minimize the number of instances where an

interesting question can't be answered by the rules in the type system.

4. CONCLUSIONS AND NEXT STEPS

Dynamically changing security policies and configurations of guard devices in cross domain environments to adjust the allowable level of information sharing is a needed capability. Solutions need to address a number of challenges, including scalability, policy representation, and C&A of dynamic policies.

The flexible adaptive policy management concepts and designs we outlined in this paper describe the core functionality by means of three functional workflows and outlines designs for hierarchical policy management agents that implement the functionality as a distributed system. We show how to use policy analysis techniques to increase the consistency of resulting policies and describe a transaction model for automatically distributing configuration updates in a robust way.

Future work will mostly centered around creation of more detailed use cases, involving policies of realistic complexity derived from real security classification guides, creation of a proof-of-concept prototype, and integration with next generation guards, such as ISSE 4.0.

5. ACKNOWLEDGMENTS

The authors would like to acknowledge the support and collaboration of the US Air Force Research Laboratory (AFRL) Information Directorate.

6. REFERENCES

[1] David Elliott Bell, "Looking Back at the Bell-La Padula Model," , Washington, DC, USA, 2005.

[2] (2009, Jan.) DISA NCES Website. [Online]. http://www.disa.mil/nces/

[3] Boyd Fletcher, XML Data Flow Configuration File Format Specification, 2008.

[4] Daryl McCullough, Marisa Gioioso, Jennifer Cormier, Carla Marceau, and Robert Joyce, "PMAF: Pedigree Managment and Assessment in a Net-centric Environment," , Orlando, 2007.

[5] Data Dissemination Tool / Workflow Enforcement System.

[6] Adam Hovak, Cross Domain SOA Challenges Workshop with CPSG, 2009.

[7] Sheldon Shapiro. (2006) An Introduction to Collaboration Gateway.

[8] A. Bradley, A. Kfoury, and I. Matta A. Bestavros, "TRAFFIC: Safe Compositional Specification of Networking Systems," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 34, no. 3, July 2004.

[9] Assaf Kfoury, Andrei Lapets, and Michael Ocean Azer Bestavros, "Safe Compositional Network Sketches: Tool and Use Cases. ," in *Proceedings of CRTS'09: The IEEE/RTSS Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*, Washington D.C., 2009.

[10] Assaf Kfoury, Andrei Lapets, and Michael Ocean Azer Bestavros, "Safe Compositional Network Sketches: The Formal Framework," in *Proceedings of HSCC'10: The 13th ACM International Conference on Hybrid Systems: Computation and Control (in conjunction with CPSWEEK)*, Stockholm, Sweden, 2010.