From ABAC to ZBAC: The Evolution of Access Control Models

Alan H. Karp, Harry Haury, Michael H. Davis

HP Laboratories
HPL-2009-30

Keyword(s):
Services Oriented Architecture; SOA; web services; access control; Federated Identity Management; FldM

Abstract:

Controlling access to resources and services is fundamental to security. A variety of access control models have been developed over the years, each designed to address different aspects of the problem. This report will examine the strengths and weaknesses of the various approaches as applied in a cross domain services and as implemented in common SOA frameworks. Please note, the access control mechanisms are discussed in this context and the comments are not general critiques of the advantages and disadvantages of the various systems. Our primary use case comes from an example investigated by the US Navy, which is examined for illustrative purposes since it is easy to understand (For more additional applicability please refer to the Department of Defense and Intelligence Community Service-Oriented Architecture Security Reference Architecture, Version 1.0 and the discussion of hierarchical policy enforcement frameworks and the section 4.2 Advanced SOAP Interaction Patterns). That discussion also extends the enclosed use case slightly to address issues it doesn't cover. Recognizing those issues led to the development of an access control model that uses authorizations presented with the request to make an access decision, an approach we call authoriZation Based Access Control (ZBAC). This paper is intended to stimulate a structured technical dialogue within the IA&A community on potential alternative enterprise approaches and possible security risks with current approaches. The KEY implementation details are in the appendices, so be sure to read them too!

External Posting Date: February 21, 2009 [Fulltext]   Approved for External Publication
Internal Posting Date: February 21, 2009 [Fulltext]

From ABAC to ZBAC: The Evolution of Access Control Models

Alan H. Karp
Hewlett-Packard Laboratories
Alan.Karp@hp.com

Harry Haury
NuParadigm
hhaury@nuparadigm.com

Michael H.Davis
SPAWAR
Michael.H.Davis@navy.mil

Abstract

Controlling access to resources and services is fundamental to security. A variety of access control models have been developed over the years, each designed to address different aspects of the problem. This report will examine the strengths and weaknesses of the various approaches as applied in a cross domain services and as implemented in common SOA frameworks. Please note, the access control mechanisms are discussed in this context and the comments are **not** general critiques of the advantages and disadvantages of the various systems. Our primary use case comes from an example investigated by the US Navy, which is examined for illustrative purposes since it is easy to understand (For more additional applicability please refer to the Department of Defense and Intelligence Community Service-Oriented Architecture Security Reference Architecture, Version 1.0 and the discussion of hierarchical policy enforcement frameworks and the section 4.2 Advanced SOAP Interaction Patterns). That discussion also extends the enclosed use case slightly to address issues it doesn't cover. Recognizing those issues led to the development of an access control model that uses authorizations presented with the request to make an access decision, an approach we call authori**Z**ation Based Access Control (ZBAC). This paper is intended to stimulate a structured technical dialogue within the IA&A community on potential alternative enterprise approaches and possible security risks with current approaches. *The KEY implementation details are in the appendices, so be sure to read them too!*

Preface

The evolving Net Centric and SOA application frameworks and the Presidential directives to share information across traditional domain boundaries requires a new approach to access control in order to implement, manage, and maintain these new systems. Fundamental to all security systems is the ability to reasonably control access to systems and data. Many initial questions and disagreements in the community have been largely semantic and differences in IA&A lexicon interpretations, partially due to the nuances in the military view of security and information assurance as compared to the commercial sector. *Additionally a significant part of the paper's thrust is contained in the appendices (A = detailed access control process / use case and B = detailed pros / cons comparison matrix), as the front materials set the stage and cover the descriptions and rationales on the proposed "ZBAC" approach*. It is recommended that individuals well versed in IA&A that already know the security risks associated therein, skim the paper for overall perspective and focus on the two appendices for a structured technical dialogue.

The fundamental premise is that identity and authentication can be cryptographically bound to all authorization steps and history. ZBAC is an attempt at eliminating the improper surrogates or authorizations that arise in these other security models in the cross domain references intrinsic to SOA and Net Centricity. It's the lack of control and accountability, especially in the service chaining issue, which is part of the broader security issues surrounding the creation of composite services in the SOA space. In essence, this report proposes ZBAC, a stateless asymmetric approach to integration, since it is the creation of secure authorizations in advance of the transactions that is the key to transitioning between domains. ZBAC also simplifies systems and user management tasks, is significantly more tolerant of intermittency, supports background pre-positioning of some security credentials and authorizations, allows chaining of systems while enforcing least privilege, and can accommodate low bandwidth constraints.

Clearly all access control must be both authentication-based and authorization-based where ZBAC continues to use the same basic IA&A elements. The change is in the pattern of use and the use of cryptographically protected transfer of authorizations between participating entities. While access control is a critical SOA IA enabler, especially with cross domain implementations, the following enterprise IA problems must also be addressed in parallel for enterprise SOA IA to work:

- use of a standard enterprise architecture and specific security standards (including versions and extensions therein),
- when SOA is deployed across domains we would expect multiple protocols and multiple data formats. So the security architecture should be able to perform given an environment that includes protocols translation and data transformation
- distributed / transitive trust model (to support the new "need to share" coalition environments),
- multi-tiered, automated delegation of privilege, authorization,
- enforcement of least privilege, other fundamental security tenets
- enhanced manageability,
- establishment of a means for secure delegation of authority between operating domains, and
- SOA SLAs, controlling overall efficiency and performance (especially on low BW / disadvantaged users).

Without a common understanding of the IA&A end-state objectives across the full spectrum of access control methods, fully addressing these enterprise problems is not possible.

Any solution must have means for handling Identity, Authentication and Authorization (IA&A). Authentication is the process of reliably verifying the identity of someone or something.[1] This is distinct from the assertion of identity (known as identification) and from deciding what authority accrues to that identity (authorization). Then, of course, the dilemma is that there are no current digital policy implementation *overarching* schemas (XACML, SecPal and WS-SecurityPolicy aside - as partial capabilities), methods to provide the authoritative sources for all those "PDPs" to provide "PEPs" a coordinated, correlated, dynamic and adjudicated decision on who gets to see what based on differences in policy, attributes and levels therein.

Additionally, IA/Security in a complex enterprise environment is also complex, relying on layers of governance, standards, processes, compliance checking, and auditing to provide adequate defense in depth. Technology alone does not guarantee IA success (witness the countless Windows systems that are left unsecured on every platform despite being given a step by step guidebook), so additional measures / processes are needed to prove the overall system assurance level, principally through the C&A process. Finally, the access control analysis conducted to date addressees DOD and recognized Federal PKI, yet until there is a recognized state, local,

[1] The word "authentication" is also used for other purposes, such as "authenticating the integrity of the contents of a message". The definition we give is a suitable one for this paper.

allied, and coalition PKI, there is some added risk for US-only NIPRNET and SIPRNET PKI supporting the CDS/MLS authentication, authorization, and web security services. Appendix C explores several observations and processes that support IA/security (including C&A) that should be on a "watch list" for all open architecture environments that stress fully operational, multi-Service integrated, automated, implementation ready capabilities.

1. Introduction

There are many aspects to building secure systems, such as privacy, integrity, non-repudiation, and others. This paper will focus on access control - deciding whether or not to honor a request for an action or series of actions. A number of models have been developed to address various aspects of this problem.

In the early days of the mainframe, when timesharing was new, people realized that the biggest need was to prevent one user from interfering with the work of others sharing the machine. They developed an appropriate access control model, one that depended on the identity of the user. Permission to use a system resource, such as a file, was linked to the user's identity. This approach is called Identification Based Access Control (IBAC) in this paper IBAC stores permissions in an access matrix, and the IBAC model doesn't include a specification of permissions for changing its entries. That left it to a trusted party, the system administrator, to make all changes. The administrator typically had unfettered rights to access anything within the system As the number of users grew, the burden on the administrator became untenable. That led to the introduction of additional concepts, such as "owner" and "group." It was hard to understand all the permissions that would be granted by adding a user to a group, and the rights granted "owner" did not allow for Mandatory Access Control (MAC).

While IBAC could manage centralized monolithic systems, distributed systems proved to be problematic for IBAC. Users could have many identities and often had to authenticate in different ways on different systems, which led to work to consolidate access control systems with Federated Identity Management, FIdM, and Single Sign-On/Single Log-Out (SSO/SLO). Managing the access rights for individuals and machines became too large a burden and prone to error, particularly de-synchronization. As complexity increased Role Based Access Control (RBAC) [1] was offered as a solution. Permissions in the access matrix were tied to roles, and which users could assume a particular role became the means of controlling user access. Challenges with RBAC became apparent when it was extended across domains. Reaching agreement with all partners on what rights to associate with a role proved to be difficult. Adding, deleting, or modifying the duties of a role involved updating too many policy stores. Further, RBAC had limited support for context, such as day *vs*. night or war *vs*. peace, when it was important in the access decision.

Attribute Based Access Control (ABAC, sometimes referred to as Policy Based Access Control or PBAC) [2, 4] or Claims Based Access Control or CBAC [3]), was proposed as a solution to these new issues. The access decision would be based on attributes that the user could prove to have, such as clearance level or citizenship. That approach made it easy to include context in the access decision. As it evolved it was also called Risk Adaptive Access Control (RAdAC).

Access implications of changing a user's attributes where left unspecified in most conceptualizations. There is also the issue of reaching agreement on the meaning of attributes when spanning organizations because of the need to reconcile complex and extensive lexicons.

IBAC, RBAC, ABAC and even CBAC – Claims Based Access Control all rely on authentication in the local accessed system context and have no native implementation of cross domain access control, although they are extensively used as the gateway to many systems. For simplicity this paper treats them as single solution and ignores those versions that have implemented aspects of the ZBAC architecture. They are collectively referred to as autheNtication Based Access Control (NBAC).[2] Section 3 explains why NBAC models lead to security problems, such as violations of least privilege and confused deputies, and management problems, such as problems updating rights, assigning responsibility, and coordinating with partners in the cross system, information sharing, and SOA arenas. As service composites and service chaining as well as broad systems integration projects continued, it became obvious that a new capability was required that could securely delegate trust and enforce the concept of least privilege.

Recognizing those issues led to the development of an access control model that uses authorizations presented with the request to make an access decision, an approach we call authoriZation Based Access Control (ZBAC).[3] This approach directly addresses the security and manageability issues inherent with NBAC.   The ZBAC model can be implemented with little change to existing systems. Further, ZBAC can be implemented entirely within the existing Services Oriented Architecture standards [5]. Even so, ZBAC is not tied to those standards. Other XML-based approaches have been considered for the Grid [6]. We have also demonstrated ZBAC using SPKI certificates [7] and without certificates for RESTful web services [8]. Other approaches have added some of the architectural elements of ZBAC in an NBAC frameworks [9].

It is also important to note that the terms NBAC and ZBAC are not as precise as desired. Many systems that rely on the user's authentication to make an access decision deliver an authorization to the service or its Policy Enforcement Point (PEP). Likewise, ZBAC systems require that the user authenticate in order to know which authorizations to grant. We have chosen these terms because they reflect what the user sees. In NBAC systems, the user submits an authentication along with the service request. In ZBAC systems, the user submits an authorization along with the request. The terms are not intended to imply more than that.

In the next section, the clear distinctions among the models are examined and how each of the models fits that framework is determined. Next, an illustrative use case is presented that makes the distinction among the approaches more clear. The use case is extended to clarify some other vulnerabilities as well. Appendices show details of the service lifecycle with ZBAC and a matrix showing various aspects of the problem and how each model deals with them.

[2] This nomenclature takes the user's view of the mechanism. Many NBAC implementations use authorizations, but the user never sees them.
[3] Our earlier papers used the acronym ABAC, which conflicts with the acronym for the more commonly used term Attribute Based Access Control.

2. Access Control

Access control is the mechanism by which services know whether to honor or deny requests. There are four pieces to the problem.

> **Identification:** Assigning a responsible party for actions. A responsible party may be a person or a non-person entity (NPE), such as a computer or a router. We'll use the term *user* to cover both cases.
> **Authentication:** The means used to prove the right to use an identity, take on a role, or prove possession of one or more attributes.[4]
> **Authorization:** The means of expressing the access policy by explicitly granting a right.
> **Access Decision:** Using some combination of the other three to decide whether or not a request should be honored.

It is common to conflate two or more of these parts of the access control problem. However, we gain a better understanding by keeping them distinct, even in a conventional system such as Windows or Unix. In such a system, identification is assigning an account for the user. Authentication lets processes prove that they are running on behalf of a particular user. Adding an entry for an identity in an access control list (ACL) is an act of authorization. Checking the ACL for a resource before granting access to it is the access decision step.

Access control becomes challenging in a distributed system, particularly one that crosses domain boundaries, such as between coalition partners, because we need to decide where and when to perform each of the steps. In a distributed multi-system environment it is only possible to identify the user in the user's local domain. No other domain possesses the context needed to identify the user. Since we expect identities to persist for some time, we do the identification step when a new user joins the domain or a new NPE is deployed. Similarly, the access decision is properly done in the service's local domain at the time of the request.

Systems based on NBAC authenticate at request time in the system's domain. The access decision is made by using that authentication to determine the authorization. Implementing NBAC in a distributed system requires that we solve a number of difficult problems, including PKI rationalization, federated identity management, and single sign on. These models are subject to a number of security vulnerabilities, such as violations of Least Privilege and confused deputy, which are described in more detail in Section 3.

With ZBAC, authorization is based on authentication in the user's domain before the request is made. The result of that authentication is one or more authorizations, which can be represented by cryptographically bound credentials or assertions such as SAML assertions. These authorizations can be submitted with a request or pre-cached for the length of time they are valid. The service or its Policy Decision Point (PDP) only needs to verify the validity of the authorization to make an access decision. The user's identity, which can be a pseudonym, can be recorded by the service for audit purposes. Appendix A shows an abbreviated example of the service lifecycle with ZBAC. ZBAC involves a relatively small change that has important

[4] The word "authentication" is also used for other purposes, such as "authenticating the integrity of the contents of a message". The definition we give is a suitable one for this paper.

implications, which are described in the next Section. The table in Appendix B summarizes the differences between typical NBAC systems and ZBAC.

3. NBAC Issues, ZBAC Solutions

Using authentication to make an access decision introduces a number of issues, which arise because the authentication is necessarily independent of the request. That separation of designation, what is being requested, from authorization, the right to make the request, opens up the possibility that the requester and the service will interpret things differently. In this section, we'll look at several of these problems.

Global Agreements

When using NBAC, the user is authenticated in the service domain, which requires prior agreement on the meaning of those credentials (and the Structure, message exchange and meaning therein). Since users invoke services in many domains, either these agreements become global or users will have to use many different sets of credentials. With ZBAC, users only authenticate in their own domains, and the connection to other systems is governed by means of an agreement between domains. The user's home or local domain can never authorize the user to do something that the local domain is not delegated to authorize.

With IBAC, the user's identity must be known to the service domain. That requires users to deal with multiple userids and multiple authentication mechanisms. Recently, Federated Identity Management (FIdM) and Single Sign-On/Single Log-Out (SSO/SLO) were introduced to address those problems, but these approaches add complexity. IBAC also is difficult to manage, requiring global updates when users enter or leave the system or have their rights change. Synchronization is a difficult task as well since users are still managed in each service domain.

ZBAC users are only managed in their own domains, which can change the authorizations they are granted when their permissions change. There is no need to federate identities or provide SSO/SLO. The ZBAC approach also protects information about the user's organization that is generally required with the identity challenges in many NBAC systems. In fact, depending on governance it is possible to allow credentials/assertions to be issued that don't even contain identity.

RBAC requires a mutual understanding of the meaning of roles. Often, a role in one domain has nearly the same rights as the corresponding role in another domain, but not exactly. The solution has been to introduce new roles to cover the discrepancies, leading to the well known problem of *role explosion*. With ZBAC, the set of authorizations being applied to a specific task is actually a calculated intersection of authorization and policy vectors, however they are encoded. Roles are one such encoding, but in this case they are defined solely by the user's domain.

Everyone must agree on a set of attributes and their meaning when using ABAC. That's not an easy task. The NSA recently spent considerable time reaching agreement on roles for use within the US Department of Defense (DoD). The participants agreed to 13 attributes, most of them related to the user’s identity, such as email address, and first, middle, and last name. Reaching agreement will be harder when dealing with coalition partners.

Attributes, such as citizenship, can be included in ZBAC authorizations to better support RAdAC. Global agreement is needed if they are to be useful, but that agreement may be easier to achieve because of the limited use of the attributes.

Excess Authority

In traditional access control systems, every program a user runs needs to be able to authenticate as the user in order to exercise any subset of the user’s permissions. This choice was appropriate in the early days of timesharing on mainframes when IBAC was introduced. In that environment programmers often ran their own programs against their own data. Today, there are three parties, often with conflicting agendas. The user is running a program written by someone else, who may have planted a back door, against data provided by a third party, who might have constructed the data to exploit a flaw in the program. Clearly, giving control of all the user’s rights to the programmer and potentially to the data provider entails considerable risk.

We see the effect of this choice daily. It is what lets a virus delete any of the user’s files and send any personal information stored on the computer to identity thieves. It also means that FIdM and SSO increase the attack surface available to malicious or erroneous software. Note that RBAC and ABAC do not solve the problem by themselves. Every program has the ability to use or abuse every permission allowed by the user’s authentication.

ZBAC encourages users to delegate subsets of their rights to programs they run. For example, editing a file requires that the instance of the word processor have access to one user file, the one being edited. With proper enforcement of Least Privilege, which is easily followed with ZBAC, a backdoor or exploited vulnerability will only be able to damage that one file.

Ambient Authority

Authentication is necessarily independent of the request being made. The result is that the access decision allows the request if any of the user’s permissions matches the request, which makes it impossible to attach specific rights to individual arguments. For example, a user wishes to copy the contents of one file to another but specifies the arguments to the copy function in the wrong order. That’s a mistake, but the user has no way to make the erroneous request fail by attaching only the user’s read permission to the input argument and write permission to the output argument.

ZBAC has no ambient authorities. Each permission being exercised is represented by an explicit authorization. Each authorization can be tied to a specific argument, allowing fine-grained control over the permissions. Combining designation with authorization in this way avoids any possible confusion over which permissions to apply.

Delegation and Revocation

Consider a user Alice with an account in a SharePoint area. Alice would like Bob to monitor one of the documents for her. With NBAC, she needs to ask Carol, the owner, to add an account for Bob and grant him access to the file. Once that is done, there is no record that Alice is responsible for Bob's access. The audit log shows that Carol created Bob's account. If Carol is unavailable, Alice can just pass copies to Bob and post his changes. No security was gained by making delegation go through Carol. In practice, the delegation process proves to be susceptible to policy degradation when Alice tells Bob her SharePoint credentials because of the trouble of doing it herself. The result of making delegation difficult is a loss of security because Bob has access to all of Alice's permissions, and Bob's identity does not show up in the audit trail. Proxy certificates were introduced into the Grid framework to address this problem. [9].

Assume that Carol set up the delegation Alice requested. Alice now wishes to undo the delegation, revoking Bob's access to the file, and asks Carol to make the change. Should Carol honor that request? There is no metadata listing Alice as the original delegator. Even if there were, Dave might have also given Bob permission. If Carol removes Bob's permission, he won't be able to do the job Dave wants him to do.

ZBAC allows Alice to delegate to Bob the exact subset of her rights she needs to get her job done. Importantly, she has little incentive to share credentials, leading to better security and auditability. Further, Bob's authorization denotes that Alice is responsible for Bob's access. That metadata is what is needed to determine her permission to request a revocation. Further, revoking one authorization does not affect any other authorizations.

Confused Deputy

Although there are a number of confused deputy attacks, such as some cross-site scripting exploits, the vulnerability is rarely called out. It arises from ambient authorities. In the classic example [10], Bob runs a compilation service that takes two arguments, an input file and an output file. Bob also keeps a log file. If Alice invokes the compilation service specifying the log as the output file, Bob's service overwrites the log with the compiler output. In many cases, there is nothing Bob can do to prevent this attack.

Confused deputy attacks fail with ZBAC. Alice uses her authorization to invoke Bob's compiler service and delegates to him permission to read the input file and permission to write the output file. Since she only has read access to the log file, an attempt to delegate write permission will not be honored, and the request will fail.

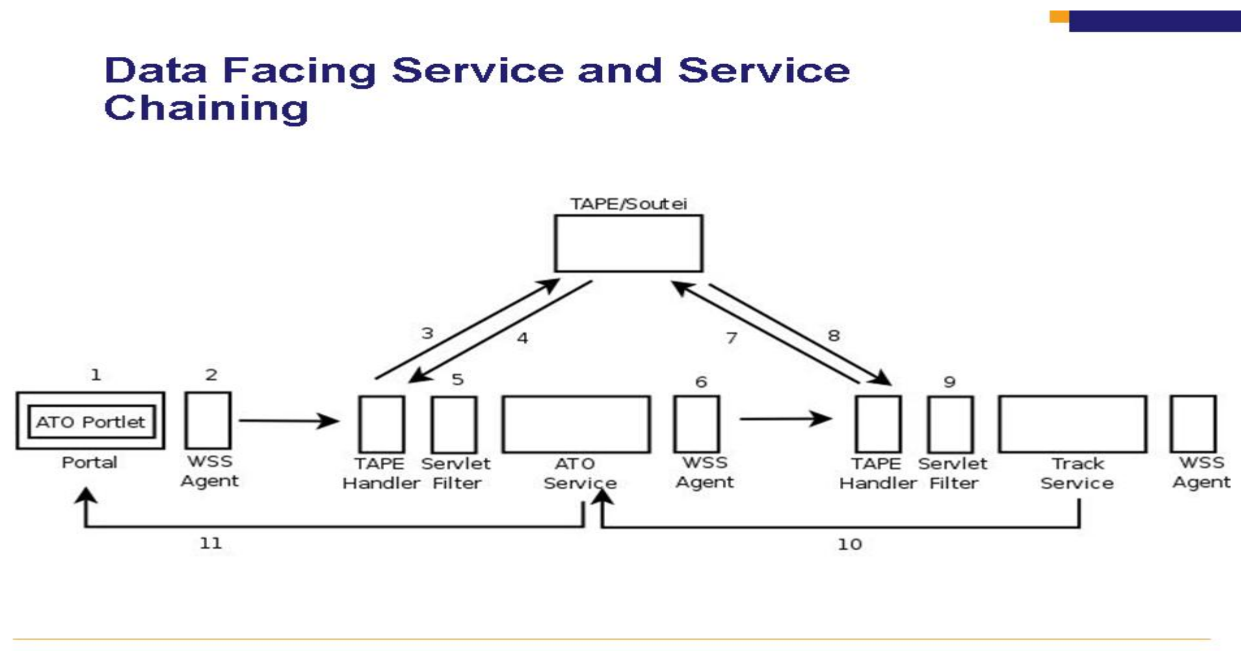
Transitive Access

A user, the invoker, invokes a service. In order to satisfy that request, the first service, the sender, invokes a second service. With NBAC, there is the question of whose credentials get used in that second invocation, the invoker's or the sender's. In many cases, neither nor both is correct. If we use the sender's credentials, then the invoker could ask for something the sender has permission to do at the second service but the invoker does not. If we use the invoker's credentials, the sender can take any action at the second service that the invoker has permission to do whether the invoker wants it done or not. The next section shows a concrete example.

There is no question of credentials with ZBAC since the rights used are explicitly represented in the authorizations. If the invoker's permissions are used to invoke the second service, the appropriate authorizations will have to be passed to the sender. Least privilege is supported because those are the only invoker permissions the sender has authority to use.

4. NAVY Use Case

Service composition is a key element in making the distributed environment useful. Assembling pre-existing services into a composite may reduce the time between identifying a requirement and implementing it from months or longer to weeks or less. In addition to the problems of protocol and data translation, managing access rights to such composites has proven to be a problem in most systems.



The figure shows the scenario covered by the NAVY Limited Technical Evaluation (LTE) [10]. The user, via the ATO Portlet, invokes the ATO Service, which in turn invokes the Track Service. The *TAPE* Handlers serve as Policy Enforcement Points (PEPs), and *TAPE/Soutei* is a Policy Decision Point (PDP).

The implementers chose to use ABAC with provider chaining from the Liberty Alliance SAML Profile [12], using *TAPE/Soutei* as the trusted third party providing the attribute assertions. With this approach, the Track Service gets a request from the ATO Service, the sender, that includes sender attributes in a Transited Provider assertion and the attributes of the user, the invoker, in an Identity assertion. The *TAPE Handler*, acting as the PEP for the Track Service, forwards these assertions to *TAPE/Soutei* for an authorization decision.

The problems described in the previous section still apply in general. To avoid them, the implementers apply this scenario with two different restrictions. One is to make the ATO Service fully trusted by the user, which defines away the risk of the ATO Service impersonating the user. However, without knowing how the TRACK Service is implemented, the user and the ATO Service must fully trust it, and so on down the chain of service invocations. Alternatively, the implementers assume that the TRACK Service only accepts invocations signed by the user, which means that the ATO Service acts as a simple router. This choice defines away impersonation and confused deputy problems at the cost of severely limiting the way the service composition can be used and by whom. However, this assumption requires that the user fully trust the TRACK Service to enforce that policy. These choices are awkward, at best, when dealing with coalition partners.

The situation is much simpler with ZBAC. The user invokes the ATO Service with the appropriate authorization. If that request includes no delegations, the ATO Service can only invoke the TRACK service with its own authorization. If the user request delegates to the ATO Service some of the user's permissions to the TRACK Service, then those are the only user rights the ATO Service can use. Least privilege is honored.

The NAVY scenario does not discuss parameter passing, in particular the passing of service references as parameters. The distinction between NBAC and ZBAC becomes even clearer when we do.

Consider a simple case that includes passing service references involving a user and two services, very much like the NAVY example. User Alice invokes a backup service provided by Bob, passing as an argument a reference to a service that will provide the data. Bob implements his backup service using Carol's copy service, which takes a reference to a service that will provide the input and a reference to a service that will hold the copy.

With NBAC, Alice's request succeeds only if Carol has permission to use both the input and output services, which is unlikely to be the case. Proposed solutions, such as provider chaining, result in Carol being able to use any of Alice's and Bob's permissions, an extreme violation of Least Privilege. Even worse, Alice's request succeeds if she specifies a service she does not have permission to use but Carol does. In other words, Alice has successfully mounted a confused deputy attack against Carol.

The situation is much clearer with ZBAC. Alice uses her authorization to invoke Bob's service and delegates to Bob permission to use the service that supplies the data. Bob invokes Carol's copy service, delegating to Carol the permission to use the input service that he got from Alice and permission to use the output service. Carol ends up with the least set of permissions she needs to carry out the request, authorizations which can be revoked when the job is completed.

There is no possibility of an inappropriate delegation in a Mandatory Access Control (MAC) environment, because the only way to delegate is over a legitimate communications channel. In a Discretionary Access Control (DAC) environment, each service's PDP can enforce delegation restrictions because it has access to the full delegation chain.

5. Summary

IBAC was introduced to prevent one user from interfering with other users on a mainframe. As the number of users grew, it became too much of a management burden to deal with all the updates when a user's permissions changed. The concept of "groups" addressed some of these problems, but not all. RBAC is an adaptation of groups to distributed systems that avoids this management problem by assigning permissions to roles and controlling which users could take on which roles. Mismatches of the rights associated with a role in different domains led to the problem of role explosion as implementers tried to make their roles more and more granular. ABAC was introduced to address those problems by providing user attributes to be used to make an access decision. An added benefit of ABAC is the way context can be combined with the user's attributes when implementing RAdAC. However, ABAC requires agreement on the meaning of attributes, and the implications of changing a user's attributes are not clear. ZBAC addresses those problems while requiring few changes to the underlying system.

ZBAC reduces the number and scope of cross-domain agreements, thereby improving scalability and reducing management overhead. By combining designation with authorization, ZBAC eliminates the kind of misunderstanding that leads to confused deputy attacks. Its delegation framework eliminates the need to manage users from other domains while simplifying the enforcement of least privilege.

RBAC addresses issues that arose when using IBAC across machines. ABAC addresses requirements not easily handled by RBAC, such as RAdAC. ZBAC addresses the needs of the services architecture, where cross domain connections are intrinsic to their implementation. Whatever comes next may require a new access control model, until then, ZBAC will more easily and securely support the full spectrum access control methods needed in a "need to share" coalition and first provider environment. It is important to understand that IBAC, ABAC, RBAC, and RAdAC are degenerate cases of ZBAC with elements missing or simplified by assumptions used in their respective designs. They still have their place when one examines the requirements of any particular systems architecture and the security risks involved.

Acknowledgements

We'd like to thank Gunnar Peterson for helping improve the quality of this report.

References

1. D.F. Ferraiolo and D.R. Kuhn (1992) "Role Based Access Control", *15th National Computer Security Conference*, October, 1992
2. M. Blaze, J. Feigenbaum, J. Ioannidis, "The KeyNote Trust-Management System Version 2", IETF RFC 2704, http://www1.cs.columbia.edu/~angelos/Papers/rfc2704.txt, September 1999
3. K. Brown, "Exploring Claims-Based Identity", http://msdn.microsoft.com/en-us/magazine/cc163366.aspx
4. A. Pimlott and O. Kiselyov, "Soutei, a Logic-Based Trust-Management System**"**, FLOPS 2006, 8th International Symposium on Functional and Logic Programming. Fuji-Susono,

Japan, April 24-26, 2006. Also in Springer's Lecture Notes in Computer Science 3945/2006, pp. 130-145.

5. J. Li and A. H. Karp, "Access Control for the Services Oriented Architecture", ACM Workshop on Secure Web Services, Fairfax, VA, November 2007, also http://www.hpl.hp.com/techreports/2007/HPL-2007-105.html
6. L. Fang and D. Gannon, "XPOLA - An Extensible Capability-based Authorization Infrastructure for Grids**",** 4th Annual PKI R&D Workshop: *Multiple Paths to Trust*, Gaithersburg, MD, April 2005
7. E-speak, http://www.hpl.hp.com/personal/Alan_Karp/espeak/Architecture.pdf, August 2001
8. Waterken, http://www.waterken.com
9. S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson, "Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile", IETF RFC 3820, http://www.ietf.org/rfc/rfc3820.txt, June 2004
10. N. Hardy, "The Confused Deputy: (or why capabilities might have been invented)", ACM SIGOPS Operating Systems Review, vol. 22, #4, October 1988
11. "TAB Response to CANES Security LTE After-action, Quicklook, Report", http://www.hpl.hp.com/personal/Alan_Karp/CANES%20Security%20LTE%20After-action%20Quicklook%20report%20-%20TAB%20input.doc 2007
12. F. Hirsch, ed., "Liberty ID-WSF Security Mechanisms Core, Version 2.0", http://www.project**liberty**.org/**liberty**/content/download/893/6255/file/**liberty**-idwsf-security-mechanisms-core-v2.0.pdf, 2006

Appendix A: Service Life Cycle with ZBAC[5]

When you look at the Information Assurance (IA) problem, the key is simplification of the dialog necessary to complete a transaction and the absolute reduction of external dependencies to the minimum possible during a transaction. Early attempts at building interoperable and services based systems used the familiar access control model that assumes everyone's identity had to be understood by everyone, that the policy encoding languages had to be common across all systems, and that this understanding had to be arbitrated in real time.

The key to creating systems that work well at scale is to remove realtime dependencies by pre-placement of appropriate credentials and authorizations, to take advantage of governance relationships to delegate and simplify the issuance and management of credentials, and to use simplified bindings to provide a provable and auditable trail to follow and interpret as required for the dissemination of credentials.

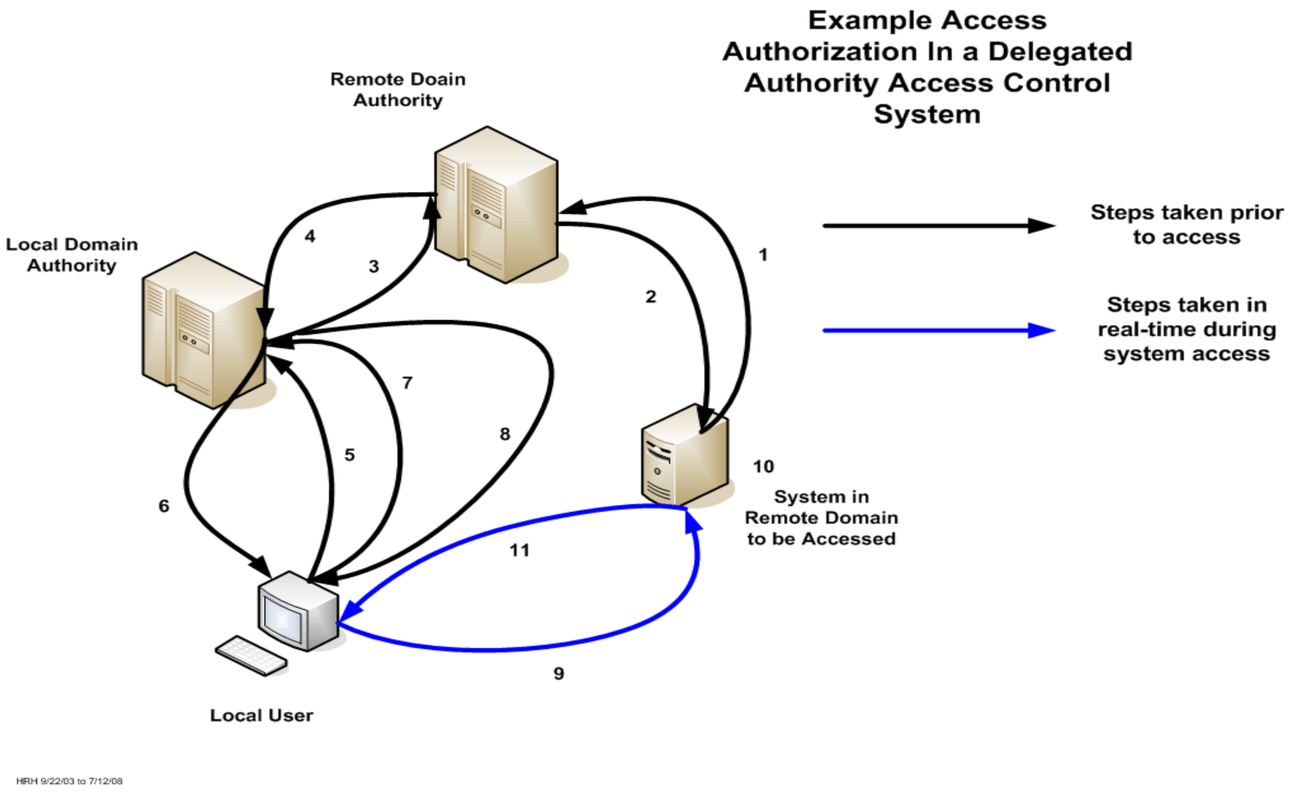

Example Access Authorization In a Delegated Authority Access Control System
Remote Doain Authority
Local Domain Authority
System in Remote Domain to be Accessed
Local User
Steps taken prior to access
Steps taken in real-time during system access
1
2
3
4
5
6
7
8
9
10
11
HRH 9/22/03 to 7/12/08

Figure A1: Simplified service lifecycle example.

An abbreviated example illustrated in Figure A1 follows:

1. Remote System accesses the Remote Domain Authority and registers identity and attributes.
2. Remote Domain Authority issues approval and credentials in the form of a certificate signed by the Remote Domain Authority's private key and containing Remote System

[5] A more complex example in the enterprise space is shown in [5].

identity information including its public key if applicable. This information allows the Local User to verify that the request is being sent to the correct service provider.

3. Local Domain Authority registers with Remote Domain Authority and agrees to MOU/Governance rules for Remote System. The Remote Domain Authority can be the root of trust for all services under its control, or can receive the appropriate authorizations directly from the services it manages.
4. Remote Domain Authority issues approval and credentials in the form of an authorization assertion or credential permitting the Local Domain Authority to issue rights delegated to it for access to the Remote System cryptographically bound to the Remote Domain Authority's private key,, and the Local Domain Authority's public key. This authorization credential may also include terms of use, expiration rules, meta-vector describing authorization, and other applicable governance restrictions on the Local Domain Authority, such as attributes that can be used to enforce RAdAC.
5. Local User registers user attributes with Local Domain Authority.
6. Local DomainAuthority issues an identity credential cryptographically bound to the Local Domain Authority's private key and the User's public key.
7. Local User requests access to the Remote System from the Local Domain Authority.
8. Local Domain Authority following relevant governance/MOU guidance issues a delegation credential to the Local User encoding the user's rights, signed with its private key and containing the public key of the Local User. The authorization credential issued by the Remote Domain Authority to the Local Authority as proof of the right to delegate is also sent in systems that have not pre-cached it.
9. Local User accesses the Remote System with a standard transaction signed by the Local User's private key, which may contain the delegation credential issued by the Local Domain Authority as signed by the Local Domain Authority's private key, and the authorization credential which was issued to the Local Domain Authority and signed by the Remote Domain Authority, (The real-time processing can be simplified here by having the Remote Domain Authority issue the credential to the Local User using the a credential issued by the Local Domain Authority designated by the authorization credential issued by the Remote System. Depending on whether the Local User has a credential issued by the Remote Domain Authority there are either two or three cryptographic operations performed here in real-time and all the keys necessary to verify the transaction are either prepositioned locally or contained in the transaction object.)
10. The Remote System:
    a. Verifies the authorization credential issued by the Remote Domain Authority using the locally prepositioned public key of the Remote Domain Authority if it is the root of trust. If the service is the root of trust, it need only verify that its own private key was used to sign the initial authorization.
    b. Verifies the delegation credential issued by the Local Domain Authority using the public key contained in the authorization credential issued by the Remote Domain Authority,
    c. Verifies the signature on the whole transaction using the public key contained in the delegation credential,
    d. Validates the assertion of rights by the Local User against the policy vector encoded in the delegation credential,

   e. Validates the authority to issue those rights against the policy vector encoded in the authorization credential, and
   f. Validates format and content of the transaction against local policy.
11. Transaction is implemented and returned by Remote System signed with the Remote System's private key.
12. The return transaction can then be verified using the Remote System's public key as found in the authorization credential.

As this example shows, parts of the service life cycle are handled differently with ZBAC. Those differences have important implications. The service creator is assumed to have full rights to the service but doesn't want to manage it, so the service creator delegates to the remote domain authority all rights to the service. In turn the remote domain authority delegates portions of its authority to the local domain authority. When local users authenticate to the local domain authority, it delegates a subset of those rights to the user. That policy decision can depend on authenticated identity, role, or attributes. When another organization reaches an agreement to use the service, the remote domain authority again delegates a subset of the service's rights to the second local domain authority. That domain controller can then delegate a subset of its rights to users in its domain. A service invocation includes the delegated authorization to use the service. The service's domain authority verifies the validity of the authorization, which can include enforcing any policy that would be violated by otherwise legitimate delegations, and sends an Allow/Deny to the service or its PEP.

Although, perhaps not intuitive from this discussion, this method of authorization has significant implications.

1. All user administration is in the local domain, eliminating the geometric explosion of permutations of user to system mappings found in many interoperable environments,
2. Allows local identity verification/authentication systems to be used intact if allowed by remote domain restrictions,
3. Eliminates intrusive reprogramming associated with layering single sign-on and RBAC on top of legacy systems,
4. Creates a rights inheritance model that can be used by some systems to automate rights management within the local domain for controlling access to remote systems,
5. Allows local groups to be used where applicable to simplify administration,
6. Authorization and delegation vectors allow precise control of privileges,
7. Repudiation and revocation checking on the user can be performed locally if PEP framework is setup for outbound enforcement,
8. Vectors are fully independent allowing changes, expirations and revocations to operate independently between systems,
9. Certificate Authorities are fully independent between domains and there is no need for users to be registered in each system or domain,
10. Nesting of the above concepts allow arbitrarily complex compositing and inheritance of rights across systems and to be chained between connected domains, which we call Federated Access Management (FAccM),
11. All assignments of rights to the user for both local and remote, are administered in the local domain,
12. Radically simplifies cryptography,

13. Limits the number of keys and certificates that have to be distributed,
14. Eliminates a lot of real time traffic and many realtime systems dependencies,
15. Allows all cryptographic operations performed in realtime to be performed locally,
16. Allows all transactions to be transport agnostic, fully stateful regarding policy and security,
17. Sets up a completely asynchronous messaging paradigm, increasing the tolerance of last mile bandwidth issues and intermittency.

Interestingly, this model also supports fine grained control of disclosure within “objects” using essentially the same transitive models. If “high assurance” object routers were implemented you would also get a significant enhancement of the controllability of the network and application connectivity by allowing routers to inspect data for content without risk of disclosure. This would also be an ideal framework for provisioning routers and controlling SLA based on QOS or COS assertions without exposing a denial of service vulnerability on the network if implemented at network entry points.

Appendix B: Authentication versus Authorization for Access Control

The following table lists a variety of issues, the IA problem related to that issue, and how it is handled by both autheNtication Based Access Control (NBAC) in its three forms that authenticate identity, role, or attributes, and authoriZation Based Access Control (ZBAC)[6]. The last column contains miscellaneous comments related to the issue.

Issue	Problem	ZBAC	NBAC	Forms of Authentication			Comments
				Identity	Role	Attributes	
Granularity	Least Privilege	LP applied to request. Each argument carries rights the user wants to apply	LP applied to "user".	User=person	User=role	User=set of attributes	Every invocation carries all user rights with NBAC
Manageability	Authentication	User only authenticates to own domain	User must be able to authenticate to all domains	Multiple logins, FIdM, or SSO	Need prior agreement on role defs, role explosion	Need prior agreement on meaning of attributes	FIdM and SSO increase rights associated with each request further violating least privilege
	Modifying rights	In response to change in user's role or service's policy, change rights given to local user, revoke as needed		Change ACLs in all relevant domains	Change ACLs in all relevant domains	Change rules in each domain's policy engine	Policy changes are not rare, NBAC needs administrator to make changes, but can overload admin
Authorization Decision	When	Prior to request	At request time				Same decision process for both
	Where	In user's domain	In service domain				

[6] The distinction is what kind of token the user presents with the request. NBAC may use authorizations, but the user never sees them.

RAdAC	Flexibility	Authorization attributes and context determine access	User attributes and context determine access				
Delegation[7]	Cooperation	Enforces policies, enables least privilege delegation [8]	Enables expression of policies that block delegation but doesn’t prevent it				In practice, users proxy or share credentials if delegation is hard
Revocation	Undoing collaboration	Right to revoke explicit in delegation, no interference with other delegations[9]	Make sure revocation request valid, doesn’t remove valid rights granted by others	may not be able to revoke login credentials	Can only remove from role	Hard to map change in rights to change in attributes	
Audit	Responsibility tracking	Delegation chain shows responsibility	Need additional metadata to track who granted a right		Need to track identity	Need to track identity	Log files impractical for tracking
Confused deputy	Vulnerability	Rights of invoker known for each argument	Can’t always distinguish rights of service from rights of invoker				ZBAC keeps designation and authorization together

[7] Delegation is only good when security is improved. Delegation in the sense meant here is not a security violation, it is an agency agreement under an MOU for a local domain controller to administer certain rights within its own, known community.

[8] The concept of delegation is that the local domain is better able to administer its local users than a remote system would ever be. Further, to the degree meta attributes are shared between systems a single characterization of a user can be used to map that user’s access in to multiple systems.

[9] The local domain authority is better able to timely revoke privileges if their users are locally under their control and since the mappings of a user to multiple systems would hopefully be in a common space whenever allowed the revocation of general rights would in effect be inherited by all the connected systems in real time

Composition	Transitive access	Rights carried with each argument	Don't know whose rights to use when first service invokes another service				Liberty Alliance Transited Provider violates Least Privilege
Trust	Global agreements	Pairwise trust relationships encoded in authorizations	Need additional metadata to track trust relations	Need FIdm ahead of time	Prior agreement on role defs	Prior agreement on attribute meanings	Trust relations hidden when authn cross domains
Identity	Coordination	Each organization uses its own approach, only coordinate form of authorization	Need global agreement on authentication	FIdM, Single Sign On, PKI rationalization			ZBAC more scalable, more flexible, easier upgrades, fewer global agreements, most pairwise[10]
PKI	Coordination	Each domain has its own CA	Need to coordinate CAs[11]				Size of CRLs a problem for NBAC[12] because of centralization

[10] ZBAC reduces the geometric explosion of permissions that must be managed for each person/entity. Revocation and repudiation are greatly simplified at any meaningful scale, and the use of local domain concepts means that authentication can be handled much faster and more efficiently on smaller access control systems having many fewer users. Due to the abstraction of the concept of authorization away from identity, the authorization of "groups" within the local domain allows simplified administration to which systems a user will automatically inherit access rights.

[11] With NBAC, the root of trust in the user's authentication token is the user's domain, so each service needs a CA to provide that domain's public key. With ZBAC, the root of trust of the authorization token is the service itself. Hence, there is no need to associate a public key with an identity on each request. Domains still need a means to identify each other when negotiating an MOU.

[12] Individual user certificates are not required in this system, only the "agency" certificates for authorization decision. So all users receive a authorization token signed by the agency certificate that defines its respective rights. The number of keys necessary is cut down significantly. This token in effect becomes a temporary certificate signed by the local domain.

Appendix C: Additional enterprise security concerns on the SOA way forward

While we addressed specific IA&A approaches in the comparison paper and appendices A and B, this is only part of a larger need to address potential systemic security concerns and methods in addressing several problematic, but required capabilities, such as of distributed / transitive trust, delegation, least privilege, manageability, system-level auditing and efficiency, to list a few. This appendix is offered as a general reference to those other items that may compound and effect the automated enterprise SOA way forward. Additionally, these IA/Security items listed below were also listed as major watch / concern items in the recent NECC TRA IRT report as areas that will need much more enterprise collaboration and coordination to fully implement an automated full spectrum access control approach in a “need to share” coalition and first provider environment.

The access control analysis conducted to date addressees DOD and recognized Federal PKI, yet until there is a recognized state, local, allied, and coalition PKI, there is some added risk for US-only NIPRNET and SIPRNET PKI supporting the CDS/MLS authentication, authorization, and web security services. For example, how can one validate IC-ISM metadata binding or authentication in a TLS dual-certificate exchange without coalition PKI? Thus, where CDS/MLS is required, the external CDS/MLS program carries the burden of implementing the external PKI of strength at least equal to the US implementation. Under web security services, and as related to the data services section, the major security methods used (including web service discovery and access decisions) should be further evaluated – ideally based on IC-ISM metadata vs. user credentials and authorization, which helps answer the "methods and processes" issue.

Scalability can also be an issue with disadvantaged low bandwidth environments and the increase in numbers of users (for example in the PKI case, added devices and services that may need credentials and privileges could significantly add to the numbers of “users” depending on how implemented (local or enterprise)), but given the current and near-term planned state of the art in IA/Security/PKI, these could be issues in implementing an automated DOD-wide, coalition enterprise environment. The dominant evidence presented is "maturity" which can hide the known dynamic nature of IA vulnerabilities, as all IT/COTS products generally have some IA related vulnerability problems, typically mitigated through defense in depth measures. (e.g., technical maturity alone is generally "yellow" concerning IA vulnerabilities - i.e., Windows 2000.)

The following observations and processes that support IA/Security (including C&A) should be on a “watch list” for all future OA programs and systems:

1. Programs are placing like communities (like COI, AHF (application hosting facilities), etc)) on the same virtualized servers. As the Programs progress (and especially as they moves to SOA), this practice may devolve and add complexity.
2. Most access control efforts are based on PKI Increment 1, where Increment 2 is supposed to support added capabilities, which should further minimize most long-term technical concerns therein; yet a few global PKI concerns to watch are:
    a) Programs are employing PKI with Representational State Transfer (REST). As the program moves to a stateless SOA and deployed SIPR PKI in the DIL, the IA interactions and resultant C&A proof could be more complex.
    b) The potential need for PKI certificates/credentials support to huge numbers of devices/services (non-person entries – NPE) creates an unclear scalability capability, which could become a CTE in future automated operational environments and readiness assessments therein.

   c) DoD PKI Increment 2 provides: (a) Tactical PKI support, (b) SIPRNet tokens/issuance, (c) HSPD-12 Compliance, and (d) Group/role hardware token capability.
   d) DoD PKI-based authentication is not yet implementable based on the non-limited availability of PKI certificates on the SIPRNet. Program users may be authenticated by a UserID/Password, using the same functions and capabilities provided in their legacy systems. Maturity of PKI on the SIPRNet significantly impacts this requirement
3. Many of the new satellites will be up by/around 2016; verbose protocols may be problematic, including the IA/Security overhead at the tactical edge.
4. Full spectrum access control supporting a "need to share" collaboration environment in a coalition and first provider response AOR is problematic in an enterprise SOA environment – principally those items list in the opening paragraph above.
5. Typically the user management function of Programs defines the access control requirements and environment at some basic level, but does not articulate the key enterprise methods and processes needed, i.e., distributed transitive trust and "security service chaining" as well as detailed use cases to demonstrate the interfaces, standards, governance, etc. required.
6. Additionally digital policy standardization, collaboration and implementation is an immature capability DoD wide, which affects the ability of enterprise Policy Decision Points (PDPs) in a mixed/shared domain environment to access an authoritative data source (for control, policy, MOU comparison and arbitration, etc). More global access control observations are;
   a) Programs typically use a synchronous validation pattern with relatively intense arbitration requirements where an ABAC or RBAC based system could eliminate most of this excessive traffic requirement if implemented asynchronously. An authorization based access control methodology, ZBAC, could resolve this and further simplify the process by supporting local delegation of authority that greatly simplifies administration and reduces system-wide key management efforts.
   b) GIG designs, because of their synchronous assumptions and high bandwidth requirements, are going to require a different approach to difficult last mile bandwidth constraints. This is an acute problem for the DIL users just from a functionality standpoint, but there is a very important architectural issue here in that the system exposed in low bandwidth last mile applications will likely be asymmetric with the core systems. This creates asymmetric IA patterns and integration patterns which can create significant emergent behavior issues with regard to end-to-end systems performance and functionality.
   c) Applications designers will then have to essentially deal with multiple object and integration patterns, depending on user context. This is generally not a good thing, but at a minimum it needs to be clearly understood and comprehensively accommodated.
7. IA/Security, and the IA controls therein, are relatively standardized throughout DOD thus they are essentially "IA equivalent" environments. Still, while Certification and Accreditation (C&A) for Programs is usually an afterthought, it should be developed in parallel to the system functions as it will be a complex, difficult, coordination and governance task, given the various combinations of different management and control schemas, new SOA/services, and interfaces to external systems.