A Shared Vocabulary for Authorization Attributes in GENI Featuring Example Policies

Ted Faber, USC/ISI

John Wroclawski, USC/ISI

Version 1.0

31 October 2011

Table of Contents

1 Introduction

This document presents a vocabulary for expressing authorization policies used by different GENI entities and explains two sample authorization policies in terms of that vocabulary. The policies are useful both for understanding the vocabulary and as basic real-world authorization policies. The policies and vocabulary described here are intended to capture discussion and emerging consensus across many parts of the GENI authorization community. Both are under active development and the authors expect to regularly revise them.

Before defining the vocabulary, we explain the problems that it is intended to solve and the requirements derived from those problems. We also briefly review the ABAC authorization framework that underlies it.

1.1 Attribute-Based Access Control (ABAC)

The GENI community is exploring the adoption of a formal logic of authentication called ABAC[1][2] and its associated cross-platform implementation called libabac[3] to share and enforce authorization policies. This system allows resource providers to explicitly state their access policies as a collection of direct and derived attributes, provides a logic engine that uses these policies to make specific decisions, and produces a record of the reasoning used to arrive at those decisions in a form readable by humans and machines.

The basic model is that an ABAC principal – e.g., a resource provider, researcher, or third party – can directly assert attributes about itself or other principals. Each asserting principal defines its own attribute namespace. In addition to direct assertion, principals can *derive* attributes in their namespaces from attributes in others using a simple, expressive, formal logic. This derivation system enables delegation and other powerful semantics.

In GENI, a resource provider decides what a researcher can do based on the researcher's attributes. The decision is based on the rules and attributes the provider knows about directly, augmented by those presented by the researcher with the request. If the provider decides the researcher has the proper attribute in the provider's attribute space, the provider authorizes the action requested by the researcher.

Each such decision is documented as a *proof*. When an access is successful, a proof explicitly describes the chain of reasoning that lead to the successful conclusion. When the access fails, the proof explains where the reasoning chain broke down. The information is presented in such a way that it, if possible, provides guidance for a new attempt with additional information that will lead to success.

Today these ideas are implemented in libabac[3], a system library accessible from many programming environments. The library encodes the rules and attributes in authenticated X.509 identity and attribute certificates. It performs access checks using the same encoding of the policy that is exposed to clients, and exports proofs that can be interpreted on multiple platforms. Simple graphical user tools are available for viewing proofs and editing policies.

1.2 Motivation for ABAC and Shared Vocabulary

ABAC enables communication of authorization policies and credentials among many independently-administered, dynamically-interoperating providers. This capability is central to the development of a federated system such as GENI, in which independent resource providers present a standard control interface while retaining the ability to implement a wide range of global and locally defined access and usage policies and agreements.

Because ABAC was designed for an environment where many players wish to collaborate within a mix of standard and customized agreements, it is very general and expressive. While this power incurs some cost in terms of processing and configuration complexity, it provides three key strengths:

- Researchers can study resource providers' policy to choose a provider that matches the researcher's needs. The policy is expressive enough to describe how a researcher can enable others to act on its behalf – a key difference between providers.

- Resource providers can define and advertise policies that can be understood without coordinating with a central authority, which makes contributing to GENI simpler.

- Endorsers or auditors can use proofs to be sure that resource providers are meeting real world commitments.

Each of these benefits arises because the various players understand the meaning of the attributes that make up the policies. ABAC enables the most general participation when all players agree on the meaning of a few attributes, but also encourages small communities to simultaneously develop their own specialized systems. This document describes a widely-understood attribute vocabulary for GENI.

1.2.1 Making Policies Understood

When a GENI researcher looks at a resource provider's authorization policy, the researcher wants to know what that provider will allow the researcher to do. The GENIAPI interface description[4] lists the standard set of operations a researcher can request from a resource provider. The authorization policy lists rules for deriving attributes in the provider's namespace. In order to determine what the researcher can do, our vocabulary makes an explicit connection between the attributes in the provider's namespace and the operations in the GENIAPI. If all resource providers make the same connection between operations and attributes – e.g., the CreateSliver attribute governs the access to the CreateSliver service – researchers can interpret any policy.

The provider has the same problem in reverse. A researcher has a collection of attributes, but they are only meaningful if the provider and the researcher agree on what they mean. In the real world, a researcher with the FootBallPlayer attribute assigned by Eton probably plays a different game than a researcher with the same attribute from Harvard; an explicit shared vocabulary avoids such problems. The more broad agreement there is on certain attributes, the more widely a policy can be applied. This vocabulary is intended to provide such a broad basis for agreement and application.

There is a trade-off between broadness of acceptance and precision of expression. Many players using the system for different objectives cannot form consensus at the same granularity as a few specialists in the same field can. ABAC embraces this. We expect specialized pockets of shared understanding (and attribute meaning) to appear, and these will not interfere with ABAC's operation in general. The global vocabulary includes attributes that can act as markers for groups that share a specialized vocabulary, which enables policy makers to detect and enforce the extended semantics.

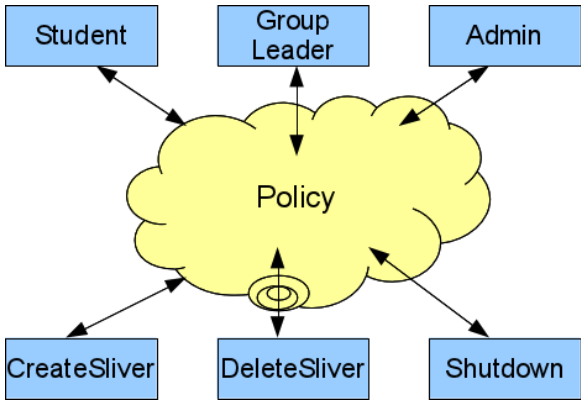

Figure 1: Policy as connection between attributes

With knowledge of the shared attribute vocabulary, a potential resource provider can *independently* construct a policy that prospective researchers can understand, and that will permit researchers to access the resources in practice.

The provider's policy can be thought of as interconnections between well-known researcher attributes and well-known operation attributes (Figure 1). Researchers and providers can interpret policies in terms of how they interconnect these well known attributes.

1.2.2 Auditing Agreements

We expect that third parties will be interested in making assertions about researchers or providers based on the agreements and behavior of those parties. For example, the GENI Program Office (GPO) may endorse providers who have agreed to allow GENI operators to shut down any sliver. This endorsement would be expressed as the GPO issuing an ABAC attribute.

While the nuances of a full legal agreement certainly exceed ABAC's expressiveness, the proofs generated by ABAC can be used to audit core aspects of those agreements. If a GENI-endorsed provider refuses to allow a GENI operator to shut down a sliver, it would be in violation of the hypothetical agreement above. Any researcher could present a proof of such behavior to an endorser.

The shared vocabulary needs to capture the attributes of interest to a provider in making generic access decisions – particularly GENI-specific attributes. It needs to capture the various GENI API operations used in resource allocation and manipulation, and it needs to express endorsement by third parties. We discuss a candidate vocabulary below.

1.3 Background Materials

The discussion below assumes a familiarity with the overall GENI API/slice-based facility architecture that underlies GENI resource allocation. There are several other documents in GENI that describe this, and we suggest at Section 3 of our earlier white paper on the SFA as a starting point.

A brief review of ABAC is included below, including pointers to the relevant literature in GENI and elsewhere.

2 Candidate Vocabulary

The shared vocabulary must be large enough to characterize researchers and facilities, but small enough that the players can all agree on the semantics. Because GENI intends to scale to many players, the candidate vocabulary is small.

The players are all ABAC principals. This includes the principals who take part in requests and responses, i.e., the researchers, infrastructure operators, the slice authorities, and the aggregate managers. It also includes parties such as the GENI program office who may wish to endorse certain other principals based on out-of-band agreements or observed behavior. A control framework or resource contributor is also a principal. While the attributes in the vocabulary may be attached to any of these principals, we encourage conventions based on the broad type of the principal.

The attributes that make up the vocabulary are all ABAC attributes. Therefore, each one is in a namespace defined by the principal defining it. In ABAC the convention is to indicate the namespace of an attribute by putting the principal name before the attribute, separated by a period. If TIED is a principal, TIED.operator is the operator attribute in the TIED namespace. If the GPO and the TIED testbed are two distinct principals, TIED.operator and GPO.operator are distinct attributes.

Attributes in the global vocabulary have the same basic meaning though their namespace can imply a scope. Continuing our "operator" example, if the operator attribute is attached to principals who operate GENI infrastructure, a principal with the TIED.operator implies an operator whose authority and responsibility are connected to the TIED testbed. A principal with the GPO.operator attribute has a broader responsibility and may be granted additional trust because of it.

Some attributes in the vocabulary are parameterized. For example, assigning a researcher a DeleteSliver attribute should imply the right to delete a specific sliver, not any sliver. We make use of single parameterized roles, a subset of the ABAC RT1 logic. DeleteSliver(sliverName) is scoped to a specific sliver, sliverName. To be explicit: TIED.DeleteSliver(sliver1) and TIED.DeleteSliver(sliver2) are distinct attributes. When the parameter must exist, but is unbound a name preceded by a question mark is used, so TIED.CreateSliver(?slicename) has a single unbound parameter. Section 3.2 describes describes the use of parameters in rules and the short and long term implementation strategies for them.

The following lays out a global attribute space and basic meanings. We group them by what kind of principal is assigning the attribute to another. Those assignments may be indirect, as Section3 describes.

We are aiming for a minimal set. Individual principals – particularly control framework principals – may define other attributes that make their policies more efficient to implement or intuitive within the framework. Groups may share specialized vocabulary as well. However, if the policies include the global attributes, a principal can ask questions about the policy in terms of the attributes in the global vocabulary and derive meaningful answers.

In terms of Figure 1, the attributes in Sections 2.1,2.2 and 2.3 are attributes with widely understood descriptive semantics at the top of the figure. The attributes in Section 2.4 are mapped to specific operations at aggregate managers and slice authorities.

2.1 Researcher Attributes

These are broad descriptive attributes assigned to a researcher by a third party or a control framework that describes their place in the GENI world. In particular, we make use of the GENI project leader and project member ideas put forth at GEC11[5]. A project leader is a researcher who is responsible for the actions taken on behalf of the members of the project. The project members are the other researchers the leader is responsible for. Projects are uniquely identified.

A researcher can have the following attributes:

Attribute	Meaning
ProjectLeader(*?project***)**	The researcher is responsible for *project*.
ProjectMember(*?project***)**	The researcher is part of *project*.
Supervises(*?principal***)**	The researcher supervises *principal*.
Student	The researcher is a student.
Operator	The principal is responsible for and controls infrastructure

The project attributes and student attribute are intended to be assigned by facilities, which as the GPO or a university. The Supervises role allows principals to directly describe their relationships to one another. The Operator role's meaning is somewhat tied to its scope, but is assigned to a person by a facility to indicate that this principal is trusted by the facility issuing the attribute to manage resources , as discussed in the example in Section 2.

2.2 Service Attributes

These roles are intended to define a servers role in the GENIAPI/SFA. Knowing that a particular principal is expected to act as a slice authority or aggregate manager simplifies exporting these roles across federated domains. There are other elements in the GENIAPI that are in some flux and not represented here. There is no registry or clearinghouse attributes. These may be added as the vocabulary evolves.

Principals acting in the GENIAPI may have the following attributes:

Attribute	Meaning
SliceAuthority	The principal is a slice authority
AggregateManager	The principal is an aggregate manager

2.3 Endorsement

A facility may issue the following role to another principal:

Attribute	Meaning
Endorses	The principal has approval of the issuer.

This is a fairly generic role assigned by one facility or other reviewing party to indicate that the given principal has the approval of the issuing principal. This is deliberately vague and dependent on the issuing principal. A GPO.Endorses attribute may indicate that the given principal follows a set of usage rules laid out by the GPO.

The Endorses attribute can also be used to mark a set of facilities or other principals that share a specialized vocabulary. The group can establish a principal to act as endorser, e.g. SpecialVocab, and then use the SpecialVocab.Endorses attribute to mark facilities that have agreed to the semantics.

One can imagine more specific attributes that indicate service levels or degrees of approval between facilities that have negotiated them. The Endorses attribute is less specific to avoid the problems of defining a global rule with byzantine meaning.

2.4 Operational Attributes

These are the target attributes for GENIAPI requests. There is an attribute for each slice authority or aggregate manager action. When a researcher requests an action the server will allow it if the researcher has the attribute with the same name in that server's name space. Concretely, if a researcher invokes the ListResources operation at aggregate manager AM, that aggregate manager will allow it if the researcher has the AM.ListResources attribute.

The aggregate manager operations are much more stable than the slice authority operations. For this table we use the current slice authority operations at ProtoGENI slice authorities. ProtoGENI is something of a de facto standard here, but this remains an area of flux. The examples here show how a mapping of attributes to operations for the current operations.

The attributes are:

Attribute	Meaning
GetCredential	The right to call GetCredential at a slice authority
GetCredential(*?object***)**	The right to call GetCredential for the given object, e.g., a slice
GetKeys	The right to call GetKeys at a slice authority
RegisterSlice	The right to register a slice at a slice authority
Resolve	The right to call Resolve at a slice authority
Remove(*?object)*	The right to call Remove on object at a slice authority
Bind(*?object)*	The right to call Bind on an object at a slice authority
Renew(*?object***)**	The right to call Renew on an object at a slice authority
DiscoverResources	The right to call DiscoverResources at a slice authority.
ListResources	The right to call ListResources at an aggregate manager
ListResources(*?slice***)**	The right to call ListResources at an aggregate manager for *slice*.
CreateSliver(*?slice***)**	The right to call CreateSliver at an

	aggregate manager, binding resources to *slice*.
SliverStatus(*?slice***)**	The right to call SliverStatus at an aggregate manager on *slice*
RenewSliver(*?slice***)**	The right to call RenewSliver at an aggregate manager on *slice*
DeleteSliver(*?slice*)	The right to call DeleteSliver at an aggregate manager on *slice*
Shutdown(*?slice***)**	The right to call Shutdown at an aggregate manager on *slice*

Any principal can assert one of these attributes, but the attribute only confers rights directly when asserted by the server being asked to carry out an operation. In the sample policies, it is common for a slice authority, SA, to assert that researcher R has attribute SA.CreateSliver(slice1) as a result R successfully registering slice1. The slice authority is asserting that it, the slice authority, is giving out the right to bind slivers to that slice. The right is only realized when an aggregate manager, AM, believes AM.CreateSliver(slice1). The example policies will show how that connection is made explicitly.

There is one additional attribute that indicates which principal made the Register call that created the slice:

Attribute	**Meaning**
Creator(*?slice***)**	The principal with this attribute created *slice*.

Many policy creators want the ability to endow slice creators with special rights to a slice, or to delegate abilities, so it seems prudent to have an agreed upon way to mark these researchers.

2.5 Slice Attributes

During discussions about the vocabulary, the idea was put forward that future iterations of the GENIAPI might operate on slices, for example connecting two slices for the purpose of sharing services between them. A special case of this would be a slice willing to transit traffic for other slices (assuming agreement about protocols and data models). We include these two attributes of slices to make that easier.

The implication is that slices are principals. This is feasible in principle, but the designers must grapple with the implications. For example, can slices act independently? If so, hat real world entity is allowed to initiate action on behalf of a slice?

The attributes are:

Attribute	Meaning
OMIS	The slice is an operations/management slice
Transit	The slice provides transit services

Before describing example policies based on this vocabulary, we review the ABAC semantics.

3 Review of ABAC Rules and Reasoning

The RT0 logic is simple and powerful, but the notation is somewhat terse. This section reviews the semantics and contrasts the RT0 expression of parameterized expressions with that of RT1.

3.1 RT0 Logic

ABAC's RT0 logic[1] allows one to attach an attribute to a principal, define a direct delegation rule and define a rule linking the possession of an attribute to the ability to delegate attributes. This section reviews the notation and semantics.

In ABAC's logic an attribute is a string attached to a principal by another principal. If an aggregate manager identified as `AM` wishes to attach the ListResources attribute to a user identified as `U` we say that `AM` has attached AM.ListResources to `U`. Only AM can assign attributes from the AM. space. Furthermore AM1.ListResources and AM2.ListResources are distinct.

There are three ways to attach an attribute to a principal:

1. **Direct assignment**. Example: U has attribute AM.ListResources. Notation:
   AM.ListResources ← U

2. **Delegation**. Example: All principals with attribute AM2.ListResources have AM1.ListResources. Notation:
   AM1.ListResources ← AM2.ListResources

3. **Linked Delegation**. Any principal P with the AM2.Linked attribute can assign the AM1.ListResources attribute by assigning the P.ListResources attribute. Notation:
   AM1.ListResources ← (AM2.Linked).ListResources

These assignments of attributes are directly equivalent to set theory inclusion statements. If each attribute is the name of a set, the three operations above can be interpreted as:

1. **Direct assignment**. Example: U is a member of AM.ListResources. Notation:
   AM.ListResources ← U

2. **Delegation**. Example: All principals in set AM2.ListResources are in set AM1.ListResources. Notation:
   AM1.ListResources ← AM2.ListResources

3. **Linked Delegation**. Any principal P in the set AM2.Linked can place a principal in the AM1.ListResources set by putting that principal into the set P.LinkedResources. Notation:
   AM1.ListResources ← (AM2.Linked).ListResources

Direct assignment is pretty straightforward. A principal binds another attribute to another principal. If we take AM.ListResources to indicate the ability to call the ListResources operation on AM, the example in 1. says that AM has explicitly granted that ability to user U.

In the second example, AM1 has expressed a rule delegating the ability to assign principals the AM1.ListResources attribute to AM2. AM2 exercises that delegation by assigning its AM2.ListResources attribute. Any principal that knows AM1.ListResources ← AM2.ListResources and AM2.ListResources ← U knows that U has AM1.ListResrources.

In some cases, AM1 and AM2 will want to coordinate closely about such a delegation, but in others AM2 may be oblivious to the delegation. If AM2 is a well known certifier or AM1 and AM2 have a general relationship where they agree on the semantics of ListResources, there is no reason to discuss the delegation. In any case, ABAC does not require any coordination to make the delegation.

The last example adds a second indirection. This delegates AM1.ListResources to a number of other principals that have an attribute assigned by AM2, not to AM2 itself. In this case a principal must know that AM1.ListResources ← (AM2.Linked).ListResources and AM2.Linked ← P and P.ListResources ← U to know that U has AM1.ListResources.

This generally allows a principal to appoint agents, other principals that will assign an attribute on its behalf. The example above showed one principal (AM1) directly delegating that authority to the agents of another (AM2). A ruleset more like AM2.ListResources ← (AM2.Linked).ListResources, AM1.ListResources ← AM2.ListResources lets AM2 express its creation of agents and AM1 delegate to the other principal. The first rule is controlled by AM2, because it controls the AM2.ListResources attribute, and the second rule is controlled by AM1 for the analogous reason.

The requirements for a delegation – the right hand side of the arrows above – can include conjunctions. For example, AM.CreateSlice ← CH.CreateSlice & SA.CreateSlice says that AM will assign the AM.CreateSlice attribute to a principal that has demonstrated that it has both CH.CreateSlice and SA.CreateSlice.

Rules for binding parameters are in Section 3.2.

Each of these operations, the assignment of an attribute or the creation of a delegation rule, is expressed in a *credential*. A credential is a signed statement of the rule or assignment in RT0 logic, signed by the principal that controls the attribute being assigned or delegated. That is, a

credential is signed by the principal whose identity is attached to the attribute on the left side of the arrow.

Such credentials form the basis of proofs in the ABAC system, and are understandable by any entity that can confirm the signatures. Libabac represents these in X.509 attribute certificates, and the examples use that format.

3.2 RT0 and RT1

The current ABAC implementation implements RT0 logic, which is described in Section 3.1 above. We expect to implement RT1 in the near future, which supports the same basic delegation rules, but with parameterized attributes. For example the AM.DeleteSliver attribute may be scoped to a single sliver by asserting an AM.DeleteSliver(sliver3) attribute.

Such parameterized attributes may be reasoned about just as unparameterized attributes are, or with some constraints on their parameters. For example, an attribute like AM.Level(4) can appear in a rule as AM.Level(?L), indicating any level is acceptable (but that an AM.Level must be assigned with a parameter) or as AM.Level(?L:[2..5]) indicating that the parameter must be in the range 2 to 5 inclusive; constraints must limit parameters to finite static sets.

When a rule has parameters on both sides, the names of the variables must bind to the same value. AM.CreateSliver(?N) ← SA.CreateSliver(?N) and SA.CreateSliver(slice1) ← U imply that U has attribute AM.CreateSliver(slice1) but no other parameterization of AM.CreateSliver(?N).

The rules for encoding current GENI access control do not rely on level comparisons, but they do benefit from scoping to a given slice or sliver name. In an RT1 world, we would express this like AM.DeleteSliver(uuid); lacking RT0 we express this scoping in the name of the attribute. For example, that attribute is AM.DeleteSliver_uuid .

Because the rules for GENI access never require arithmetic or other operations, just matching, and only principals who issue scoped attributes need to interpret them, we can guarantee that no confusion results.

4 Sample Policies

We define two policies for a GENI facility using the vocabulary above. The first is fairly simple primarilyto show how to express policy in ABAC, the second incorporates more complex delegation.

In both cases we define a policy for the TIED facility, which consists of a slice authority, SA, and an aggregate manager AM. Each of these are represented as ABAC principals (TIED, SA, and AM). There is also a well-known principal representing the GENI Project Office, GPO. We will also mention principals PL, a project leader, and PM a project member.

We present the policy first in English and then the ABAC rules that represent the policy. The rules are each encoded into a credential and can be distributed in various ways throughout TIED.

4.1 Simple Policy

The simple policy allows any GENI project leader to create a sliver at the TIED slice authority, and allows any researcher who has been issued a standard CreateSliver(slice) credential from a GENI-endorsed facility's slice authority to create and manipulate slivers at the aggregate manager. TIED itself is a GENI-endorsed facility.

4.1.1 Endorsement Credentials

The GPO principal expresses its endorsement of TIED by issuing a credential to TIED:

- GPO.Endorses ← TIED

Similarly the GPO principal recognizes PL and PM as leader and member of project p by issuing[1]:

- GPO.ProjectLeader(p) ← PL
- GPO.ProjectMember(p) ← PM

4.1.2 Facility Credentials

TIED advertises the roles of its principals by issuing credentials that make their responsibilities explicit. That is the TIED facility declares SA to be a slice authority and AM to be an aggregate manager by issuing

- TIED.SliceAuthority ← SA
- TIED.AggregateManager ← AM

4.1.3 Slice Authority Policy

Because the policy allows a GENI project leader to take any action at the slice authority, most of the roles have the same format:

- SA.GetCredential ← GPO.ProjectLeader(?)
- SA.GetKeys ← GPO.ProjectLeader(?)
- SA.RegisterSlice ← GPO.ProjectLeader(?)
- SA.Resolve ← GPO.ProjectLeader(?)
- SA.DiscoverResources ← GPO.ProjectLeader(?)

[1]This direct assignment is the simplest assignment. More scalable would be to issue GPO.ProjectLeader(p) ← PM and GPO.ProjectMember(p) ← (GPO.ProjectLeader(p)).ProjectMember(p). That allows PL to add members to project p by issuing a credential like PL.ProjectMamber(p) ← PM, without involving the GPO principal.

The most interesting of these for this example is SA.RegisterSlice. When PL registers a slice, called slice1 here, SA issues it credentials encoding the following attributes:

- SA.Creator(slice1) ← PL
- SA.GetCredential(slice1) ← PL
- SA.Remove(slice1) ← PL
- SA.Bind(slice1) ← PL
- SA.Renew(slice1) ← PL
- SA.CreateSliver(slice1) ← PL

The first recognizes PL as the creator of slice1. This policy does not make use of that fact, though others may. The last asserts that SA believes that PL can create slivers attached to slice1. The others allow PL to invoke slice authority operations on SA that modify or report on slice1.

In addition to the list above, SA will also return the following credentials that were described in Section 4.1.1 and 4.1.2 to describe its own capabilities in the global vocabulary:

- TIED.SliceAuthority ← SA
- GPO.Endorses ← TIED

4.1.4 Aggregate Manager Policy

The first elements of AM's policy allow project leaders to list resources generally and for a GPO or TIED operator to shutdown any sliver

- AM.ListResources ← GPO.ProjectLeader(?)
- AM.Shutdown(?) ← GPO.Operator
- AM.Shutdown(?) ← TIED.Operator

The aggregate manager will recognize CreateSliver credentials from any slice authority from a GPO-endorsed facility. Two rules accomplish this, one to recognize the set of slice authorities that AM trusts, and one to connect their authority to the ability to create slivers.

- AM.GPOSliceAuthority ← (GPO.Endorses).SliceAuthority
- AM.CreateSliver(?slice) ← (AM.GPOSliceAuthority).CreateSliver(?slice)

Once an AM creates a slice, it issues credentials allowing further operations (we use PL for the principal who created the sliver).

- AM.ListResources(slice1) ← PL
- AM.SliverStatus(slice1) ← PL
- AM.RenewSliver(slice1) ← PL
- AM.DeleteSliver(slice1) ← PL
- AM.Shutdown(slice1) ← PL

4.1.5 Example Proofs

The two most complex authorizations under this policy are the right to create a slice at SA and the right to create a sliver at AM. We show those authorizations.

For PL to create a slice at SA, SA needs to prove PL has attribute SA.RegisterSlice, or that PL is a member of SA.RegisterSlice, depending on which formulation is clearer. PL presents a credential encoding GPO.ProjectLeader(p) ← PL. SA uses the rule SA.RegisterSlice ← GPO.ProjectLeader(?project) to show that because PL has GPO.ProjectLeader(p) it has GPO.ProjectLeader() for some ?project, and has SA.RegisterSlice.

The reasoning at AM when PL requests CreateSliver(slice1) is more complex. PL presents credentials encoding SA.CreateSliver(slice1) ← PL, TIED.SliceAuthority ← SA and GPO.Endorses ← TIED with its request. It selects those based on AM's published policy. If AM's policy is unpublished or partially published, PL would include all the credentials it received when it successfully registered the slice.

AM is trying to prove that PL has AM.CreateSliver(slice1) because it called CreateSliver with a slice parameter of slice1. AM reasons that SA is in the set AM.GPOSliceAuthority, because TIED is in GPO.Endorses (GPO.Endorses ← TIED above) and SA is in TIED.SliceAuthority and the rule AM.GPOSliceAuthority ← (GPO.Endorses).SliceAuthority applies.

Similar reasoning applies to get to AM.CreateSliver(slice1) from SA.CreateSliver(slice1) and the fact proven above that SA is in AM.GPOSliceAuthority.

Other proofs are more straightforward.

4.2 Policy With Delegation

We extend the policy from Section 4.1 with two delegation points. First we allow either a project leader or project member to invoke commands on SA, including RegisterSlice. Second we allow a slice's creator to delegate the authority to make any of the aggregate manager calls on the new sliver at AM. These policy changes are all internal to the TIED facility, so the credentials issued by the GPO in Section 4.1.1 remain unchanged, as to the facility credentials described in 4.1.2.

The delegation of sliver rights occurs by the slice creator assigning the operational attribute to the delegatee directly. If the creator of slice1 is C and it wants to allow D to create slivers attached to slice1, C will issue:

- C.CreateSliver(slice1) ← D

4.2.1 Changes at SA

All the SA credentials described in 4.1.3 remain in force. A project leader can continue to to all the things they could under the policy in Section 4.1. To allow project members to operate at SA, we add the following:

- SA.GetCredential ← GPO.ProjectMember(?project)
- SA.GetKeys ← GPO.ProjectMember(?project)
- SA.RegisterSlice ← GPO.ProjectMember(?project)
- SA.Resolve ← GPO.ProjectMember(?project)
- SA.DiscoverResources ← GPO.ProjectMember(?project)

4.2.2 Changes at AM

The first changes at AM are to allow the creator of any slice authority from a GPO-endorsed facility to be able to create a sliver attached to that slice. The following rule does this, reusing the AM.GPOSliceAuthority ← (GPO.Endorses).SliceAuthority rule that identifies trusted slice authorities.

- AM.CreateSliver(?slice) ← (AM.GPOSliceAuthority).Creator(?slice)

To handle delegation, AM adds two rules. The first identifies the creator of a slice from a slice authority in a GPO-endorsed facility, and the second accepts the delegation for CreateSliver.

- AM.Creator(?slice) ← (AM.GPOSliceAuthority).Creator(?slice)
- AM.CreateSliver(?slice) ← (AM.Creator(?slice)).CreateSliver(?slice)

The first rule in this section could now be rewritten as:

- AM.CreateSliver(?slice) ← AM.Creator(?slice)

To allow the creator and the creator's delegatees to call the remaining operations, AM must add:

- AM.ListResources(?slice) ← AM.Creator(?slice)
- AM.ListResources(?slice) ← (AM.Creator(?slice)).ListResources(?slice)
- AM.RenewSliver(?slice) ← AM.Creator(?slice)

- AM.RenewSliver(?slice) ← (AM.Creator(?slice)).RenewSliver(?slice)
- AM.DeleteSliver(?slice) ← AM.Creator(?slice)
- AM.DeleteSLiver(?slice) ← (AM.Creator(?slice)).DeleteSliver(?slice)
- AM.Shutdown(?slice) ← AM.Creator(?slice)
- AM.Shutdown(?slice) ← (AM.Creator(?slice)).Shutdown(?slice)

5 Summary

This document has presented the current ABAC vocabulary for cross-federation authorization in GENI, as well as two sample policies that use that vocabulary. We believe that the combination of discussion and examples shows the power of the pairing of the small set of shared attributes and ABAC's reasoning.

A demonstration of the policies above using the libabac implementation will be shown at GEC12.

References

[1]Ninghui Li, John C. Mitchell, and William H. Winsborough, "Design of a Role-Based Trust Management System," *in Proceedings of the 2002 IEEE Symposium on Security and Privacy*, (May, 2002).

[2]"ABAC Authorization Control Model and Discussion", *http://groups.geni.net/geni/wiki/TIEDABACModel*, 2011

[3]LibABAC Home Page, *http://abac.deterlab.net*, 2011.

[4]The GENI API, *http://groups.geni.net/geni/wiki/GeniApi*, 2011

[5]Aaron Falk, "Federation in GENI", GEC11, 2011 *http://groups.geni.net/geni/attachment/wiki/GEC11Federation/GENI%20Federation%20-%2022July2011.pdf*