Authorization in GENI

March 14, 2011

Rev 0.4

Contributors
Stephen Schwab, SPARTA
Ted Faber, USC/ISI
Rob Ricci, Utah
Jeff Chase, Duke
Aaron Helsinger, BBN
Tom Mitchell, BBN

Introduction

This is an initial sketch of the next steps towards getting ab Authorization in GENI strategy worked out for the GENI AM API. The basic idea is to indicate what ProtoGENI and ORCA each need to use ABAC as the authorization approach within GENI, and then to roughly schedule the order of analysis, development, integration and test deployment activities to make this happen.

High Level Strategy

- Write document laying out design & implementation plan (Steve Schwab, remainder of this document)
- Present Plan & Schedule at GEC 10 (Steve Schwab and Ted Faber)
- Seek community feedback and consensus at GEC10 (all authors)
- Test and field integration with ProtoGENI in the GPO lab (ProtoGENI, GPO, ISI, SPARTA) and independently with ORCA (ORCA/BEN, other cluster projects.)

Basic Approach

Two control frameworks will be integrating ABAC assertions into their implementations as an authorization scheme for GENI. ProtoGENI and ORCA will each execute a plan to use attributes for authorization.

ProtoGENI will adopt ABAC credentials as a second format on the wire, using the current libabac implementation available from ISI (http://abac.deterlab.net). Under this approach, an implementation may pass ABAC credentials as an alternative to the current native ProtoGENI credentials. Authorization calls will be made on the server side to the libabac engine to verify that credential assertions satisfy the policy required to authorize (affirmatively permit) the requested operation to be invoked. During a transition period, both native and ABAC credentials will be supported, to assess usability and maturity of implementations.

ORCA will adopt ABAC credentials in a manner consistent with the Slice-Based Facility Architecture abstractions. In particular, ORCA will leverage the *owner, delegateOwner* and *speaksFor* attributes in a stylized fashion to encode ownership over a resource and delegation of specific or collections of rights to manipulate or invoke operations that are mediated by ABAC authorization checks.

Detailed Plans

ProtoGENI with ABAC: Concept of Operations

1. Use of a single round of ABAC negotiations as the default supported case for the immediate future. Defer use of multiple round negotiations until required by a GENI use case. *Multiple round negotiations are primary useful to protect sensitive attributes from disclosure in settings where privacy is a requirement.*
2. Users acquire ABAC credentials to use ProtoGENI – sketch out life cycle, end-to-end process. Which assertions are generated by client tool, user registry, or slice authority?
3. AMs get a default/prototypical "AM policy" from ProtoGENI – sketch out this policy, and how it will be maintained and distributed. All participants must come to an agreement on the set of assertions – the standard vocabulary -- to be used for the local default AM policy. (ProtoGENI/ISI/SPARTA will craft a workable AM policy as a starting point..)
4. Generate and agree on the ABAC assertions for users and slices to permit operations at the local AM. (A one-to-one policy mapping between attributes and API calls is the simplest candidate. More complex policies, such as one encoding the current GENI authorization semantics within ABAC are defined in a related GENI Rules in ABAC document by Ted Faber and John Wroclawski.)
5. AMs may tailor the AM policy to reflect local policy issues. For example, certain pools of components within the aggregate may be reserved for local classes or research projects. Such a policy could be expressed by reference to explicit local attributes.
6. ABAC supports multiple sources of user assertions (credentials) as well as multiple sources of AM (provider side) policies. It is the union of all credentials that determines the outcome of an authorization check as valid or not valid.
7. Security Binding Property, as an abstraction to introduce, define and discuss further. For ProtoGENI, only the ID associated with the credentials (ABAC assertions) can use those assertions to invoke an operation. ORCA may take a less restrictive view on this, to be discussed further under an expanded ORCA Concept of Operations.
8. Discuss long-term plans, including the possibility of adopting a dual-credential scheme, or transitioning to exclusive use of ABAC.
9. [DEFERRED FOR FUTURE DISCUSSION] ABAC and InCommon/Shibboleth interoperation could be achieved using *identity portals*. The near-term approach articulated in this document defers discussion on this topic, but the design and implementation will be undertaken with awareness of this branch of work, to ensure the long-term ability to leverage these additional sources of attributes is not precluded.

ProtoGENI development, integration and deployment

1. Development task: tool for specifying AM policy (version 0.1 [GEC10]; version 1.0 [GEC11])
2. Development task: tool for examining AM policy (version 0.1 [GEC10]; version 1.0 [GEC11])
3. Development task: standalone tool for specifying User assertions (version 0.1 [GEC10]; version 0.5 [GEC10 + 2 months])
4. Integration task: integrate tool functionality for specifying User assertions into clients (omni, others) (version 1.0 [GEC11])
5. Test Use Case: Perl wrapped libabac sample code (ISI, [GEC10])
6. PG tests directory – select test cases for create slice, create slivers on a slice, etc. – to figure out what we need to adapt for PG + ABAC test cases
7. Analysis task: Build, run, evaluate current libabac sample test cases in environment similar to ProtoGENI (PG, [GEC10])
8. Analysis task: How does a client determine that ABAC assertions are accepted at an AM? (Do we need a "credential versions accepted" call?)
9. Analysis task: Inside the Slice Authority, ProtoGENI's older Emulab code makes calls to local_root() to verify authority to create a slice. How will this call/policy check be accommodated when using the libabac implementation? (PG, SPARTA [GEC10])
10. Analysis task: The current ProtoGENI credentials use a complex two level mapping scheme from privileges to types to operations. Replace with a one-to-one ABAC asserted privilege to API-level fine-grained policy – basically, if a user has the privilege, they may invoke that API-level call. (ISI [GEC10])
11. Integration task: Modify ProtoGENI AM sources to invoke libabac for authorization checks. (PG, ISI, SPARTA)
12. Development task: Need to work out how to scope to include the "object identifier", e.g. slice ID or other specific ID within an ABAC assertion. (version 0.1 [GEC10] Implementation approach is to use attribute name-space overloading with RT0 initially; version 1.0 [GEC10 + 2 months] libabac implementation extended to support single explicit parameter. Note: we believe we only need support for a single RT1 parameter, which should accelerate implementation.) (ISI, SPARTA)
13. Development task: Assertions have lifetimes (credential timeouts). ProtoGENI needs to be concerned with assertion lifetime and how to support certification revocation lists (CRLs) at the rate of about 1 CRL update / 24 hour period. (PG, [GEC11])
14. Development task: Additional Tool: ProtoGENI flash clients parse the current native credentials in XML format. We need to create a webservice (to be run over the network, or better yet, locally) that will convert ABAC-encoded-credentials into renderable XML so the flash clients (and any other similar client) can parse-and-display via a call on the webservice. (ISI, SPARTA, [GEC10 + 3 months]
15. Lab Test Case: GPO lab to build, configure, exercise end-to-end user-slice creation lifecycle using ABAC authorization scheme as-integrated in PG.

16. Field Test Case: GPO lab as a source of third party user assertions and third party AM policy assertions.

ORCA with ABAC: Concept of Operations

1. **Basic ABAC definitions:** Subjects have attributes. Possession of an attribute says the subject is a member of a set associated with some role. Subjects may empower servers to act in their behalf, by delegating attributes to those servers. Attributes are asserted by attribute roots. Any entity can be a root for its own name space.
2. Objects have attributes. An authorization policy for an object may consider these attributes. One attribute that an object may have is an object name. Subject attributes may have a single parameter: an object name. An authorization policy for an object may filter on the subject parameter.
3. When a subject attempts to operate on an object, an authorization policy determines if the operation is permitted. The authorization policy may consider attributes of the subject, the object and any relevant ABAC inference rules. The policy knows the sources of those attributes and inference rules (i.e., the roots), and it considers those as well.
4. **GENI Semantics:** The subjects are users (experimenters) and operators, and software operating on their behalf. Objects are slices. There are one or more IdPs that authenticate users and operators and endorse their public keys. IdPs issue GENI credentials to subjects. IdPs act as ABAC roots: GENI credentials are ABAC credentials.
5. Users create slices, or request them to be created. The entity that approves slices is called an SA. Users become owners of their slices.
6. Owners may delegate ownership privileges on their slices to other users or groups.
7. Users request resources for their slices from AMs. AMs assign resources to slices. An assignment of resources to a slice is called a sliver.
8. AMs advertise themselves and their resources to one or more clearinghouses. Users may request a clearinghouse to identify suitable AMs, and resources available through those AMs.
9. **Instantiation of a slice:** User obtains GENI credentials from one or more IdPs. User requests SA to create a slice. The SA assigns a name S to the created slice, and endorses it, (e.g., issues SliceID.) The SliceID may include attributes of the slice, (e.g., "SA says this is a GENI slice".) The SA issues an ownership attribute for the slice, "SA.owner(S)<--user".
10. User requests AM to create a sliver for slice S, and assign resources to it. User requests a ticket for resources at AM. User requests the AM to create the sliver, passing the ticket, if any.
11. **ORCA specific abstractions:** Users act through slice managers (SMs). A user may use ABAC to delegate some or all of its attributes to an SM that is speaking for that user. AMs may delegate to CHs some of their power to assign resources to slices. The tickets may be requested from a broker/CH. This is a policy decision point: the broker/CH may consider user credentials, generally the same credentials that an AM might require. The CH must be pre-empowered by the AM to issue the ticket. This occurs outside of ABAC, using SHARP ticket delegation.

Additional Notes

1. ORCA (GENI) credentials may (will) be retrieved from a number of different services, including an identity Portal, or idP, that the user authenticates to at their institution. They may use any identity or authentication scheme, including ones administered by their local campus, to retrieve these credentials.

2. Attributes associated with an ABAC identity may be made in credentials (assertions) provided by any source of such attributes – whether internal to the GENI eco-system or an external source. Users may collect and forward sets of these credentials as required to gain access to specific AMs or slices.
3. User credentials (attributes), and not slice credentials or attributes, are the sole means for the requestor (subject) of an operation to provide input the authorization check. (This authorization check may be referred to as the policy decision point, or PDP, in the ORCA design documentation.)
4. AMs and Slices will be the primary objects on which operations are invoked, and hence against which authorization checks will be performed. In practice, this means that ABAC authorization policies will seek to establish whether a set of credentials entails the possession of a specific attribute (right or privilege) relative to an AM or slice, but the root of this authorization policy namespace will typically be neither the AM or slice, but the SA. (Delegation will be used by the AM's policy to grant the SA the ability to assert these types of policies.)
5. Hierarchical attribute delegation is subsumed by the more general ORCA (SHARP)ticket delegation abstraction. (ORCA's ticket mechanisms are strictly more powerful than ABAC assertions, in that the mechanism may be used to delegate claims on resources that go beyond authorization policy to include resource management decisions.)
6. The owner of an ORCA slice may delegate (via ABAC assertions) the rights to perform a subset of operations on that slice to another ORCA principal.
7. Authorization points (PDP points) in ORCA must be configured with the list of anchors – roots in ABAC – that are trusted and permitted to issue credentials with attributes about ORCA requestors (subjects).

ORCA development, integration and deployment

1. Analysis: Build and install libabac for use in the ORCA development environment.
2. Analysis: Determine how to call libabac from Java.
3. Development: Modify SM to act as an SA: generate a self-signed SliceID credential for each new slice.
4. Development: Modify SM to act as an SA: generate a slice ownership cert for the user.
5. Development: Modify SM to tack credentials onto requests as a property.
6. Integration: Modify AM to call libabac to validate credentials passed with request.
7. Trial Deployment: Configure libabac on AM with suitable anchors and inference rules for authorization policies.
8. Development: Modify AM to call libabac to check request and credentials against authorization policy.
9. Development: Enhance error logs to handle authorization failures, and possible retry.