Proposal: Use of URN's as GENI Identifiers (Version 0.3, Draft)

Camilo Viecco - Senior GMOC Programmer

Introduction

Identifiers are labels used within a system to uniquely name a particular entity. In GENI, the use of *GENI Global Identifiers* (GGID) has been proposed to be the the identifiers for entities within GENI. GGID's are described by three documents in the GENI documentation: The Slice Facilities Architecture [1], the GENI identity control framework document [3] and the GENI Facility Design [2].

GGID are presumed to provide a foundation for a '*correct and secure system*'[2]; however as defined in the GENI documentation they neither form persistent identifiers nor provide provable authentication. This document tries to document these problems and propose a different approach for identification of GENI resources.

Motivation

GGID, as currently defined, are fundamentally flawed as they have been given two distinct and incompatible functions. GGID are supposed to both identify an object (or resource) and authenticate the object (or resource). To illustrate this, lets take a look at the definitions of identifiers and GGID in the current GENI literature:

From the Slice Facilities Architecure Document[1] (section 4):

The SFA defines global identifiers (GID) for the set of objects that make up the federated system. GIDs form the basis for a correct and secure system, such that an entity that possesses a GID is able to confirm that the GID was issued in accordance with the SFA and has not been forged, and to authenticate that the object claiming to correspond to the GID is the one to which the GID was actually issued.

Specifically, a GID is a certificate that binds together three pieces of information:

GID = (PublicKey UUID[1]*, Lifetime)*

From the Facilities Design Document[2] (section 2.2.1):

The GMC defines unambiguous identifiers—called GENI Global Identifiers (GGID)—for the set of objects that make up GENI. GGIDs form the basis for a correct and secure system, such that an entity that possesses a GGID is able to confirm that the GGID was issued in accordance with the GMC and has not been forged, and to authenticate that the object claiming to correspond to the GGID is the one to which the GGID was actually issued.

From the identity Document[3] (similar rationale than [2]) :

1 **This footnote is NOT present in the original text**. UUID are 'Universally Unique Identifiers' and their generation is defined in both a ITU recommendation [7] and the functionally equivalent RFC 4122 [8]. UUID are 128 bit numbers that if generated cooperatively have a very low probability of clashing.

An identity, to my mind, has 2 simple properties:

If a principal asserts an identity, anyone in the system can confirm that directly with the principal. That is, I can prove that I am who I say I am.

Each identity maps to one principal, so two elements in the system with information tied to the same identity are sure it's information about the same principal. That is, identities are impossible to forge. (Or impossible to forge with very high probability).

While the specifics of the implementation of this ideas is incompatible, the really troublesome issue is the fact that identity and authentication are being bundled together. These ideas are incompatible because an identity is nothing more than a *persistent* label on an object while authentication is the process of validating some information that presumably was created by a certain object (more explicitly confirming the identity of a person (or entity), or the origins of an object). Simultaneous authentication and identification are incompatible because authentication is the validation of a mapping between an identity and some other object (usually a public key or a token).

GGID are also problematic because, as currently defined, they limit the mechanisms and techniques used by entities within GENI. In general, two techniques are used to prove identity in remote system: the use of shared secrets and the use of public key cryptography. When using shared secrets authentication is done by proving to the authenticating user that the authenticating party knows the shares secret. Also the protocol to prove this should: prevent the any third party from gaining much knowledge about the secret and prevent replay attacks.

Authentication via via cryptographic techniques involves at least a three step process. First, the entity that wants data to be authenticated signs the authenticated data with its private key (we are assuming that this entity has generated beforehand a public key pair). Second, the entity authenticating the object must acquire in a trustworthy manner the public key of the signing party. Third, the authenticating object validates the signature of the data with the now know public key. Now, assuming that the cryptographic primitives are secure, the what the authenticating entity validates is that its the public keys of the signed data. What makes the authentication work is the trustworthy mapping (from the authenticating party perspective) of the entity identity to the public key.

Authentication however does not necessarily require for the authenticating entities to prove directly to each other their identities. It is also common to use a trusted third party by the authenticating party to authenticate users. This trusted third party knows how to authenticate the user/entity.

GGID are currently defined as X.509[9] certificates [2]. Thus we are not only limiting the mechanisms to to authentication but we are also binding the technology do to authentication. What we need is to define a identity scheme to be used by GENI and a recommended technology to to authentication. The discussion of technologies for authentication is out of the scope of this proposal

Why not use UUID instead?

The previous discussion would can lead to a simple solution (hinted by the GENI Facility Design Document). Use UUID as the GENI identifiers and keep the GGID not as identifiers but as the authentication technology for GENI. From the Facilities Design Document[2]: *Specifically, a GGID is represented as an X.509 certificate [X509, RFC-3280] that binds a Universally Unique Identifier*

(UUID) [X667] to a public key.

While this solution might seem to satisfy most of the issues addressed above several problems several problems exist.

1. UUID are neither can be neither unique or universal. A malicious user (or a subverted system) can generate clashes in the UUID name-space. There is no way to specify hierarchy in the UUID name-space.
2. GENI would require a correct high availability complete registry. Policy rules are usually specified in terms of human readable names, therefore each authorization procedure would need to check the GENI registry at each authorization decision. Also this registry must be correct and updated so that all this requests can be completed correctly. This registry is another single point of failure of the system. If someone can impersonate the registry, it can impersonate any GENI entity.
3. The registry name (as defined) uses the same separators for GENI hierarchy as what is commonly used in the Internet for administrative hierarchy. This is problematic as it makes necessary for GENI to make names for each node even with nodes have already name well known name bindings.
4. Complicates auditing. Accurate auditing of GENI would either require a historic registry or auditing both the UUID and the human readable name as reported by the registry.
5. Prevents inter-operation with other authentication mechanisms. No other commonly used authentication system used UUID as user identifiers. Therefore even if we abandoned GGID as being certificates inter-operation would require the addition of another trusted mapper from UUID to the interface naming scheme.

So using UUID solve part of the GGID problem but it still very problematic. The UUID to human readable name mapping complicates the system and does not provide any useful features. What is needed is a form of human readable and compatible identifiers.

Proposal

Human readable names simplify interpretation and there are many standards to define human readable names. For GENI we are proposing to use two types of identifiers: Uniform Resource Names (URN) [4] and email addresses. URN can be used to identify both resources and users, email addresses are only used to identify users. For URN we also propose using the publicid URN namespace [5], using IDN. This namespace is suggested as using it does not require any registration of any namespace with IANA.

The rationale for this proposal us that URN provide a well established, standarized, extensible and human readable name-space in which arbitrary object can be made. Further URN provides implementation specific details such as the valid character set and encoding of the names.

URNs also provides a mechanism to encode other non-geni identifiers so that interafacing other systems does not require additional mappings.

For encoding URN we first transform the resource name into a string that can be then transformed into a URN via the publicid encoding process. We suggest the following naming convention (in perl regular expression syntax for the encoding string:

```
"IDN [toplevelauthority][\/\/sub-authority]* [resource-type] resource-name"
```

This would lead to the following urn schema in the public id namespace (Using the transformation in [5]:

```
"urn:publicid:IDN+toplevelauthority[:sub-authority]*[\+resource-type]\
+resource-name".
```

Examples of the transformations can be seen in table 1.

Resource	Identifier Type	GENI Identifier
User cviecco at the planetlab namespace	urn:	urn:publicid:IDN+planet-lab.org+user+cviecco
User cviecco at the iub.edu delegation in the planetlab namespace	urn:	urn:publicid:IDN+planet-lab.org:iub.edu+user+cviecco
Planetlab node: pl2.ucs.indiana.edu	urn	urn:publicid:IDN+planet-lab.org+node+pl2.ucs.indiana.edu
Interface eth0 in planetlab node pl1.ucs.indiana.edu	urn	urn:publicid:IDN+planet-lab.org+pl2.ucs.indiana.edu:eth0
Internet2 circuit: I2-ATLA-JACK-I2-05128	urn	urn:publicid:IDN+internet2.edu+circuit+I2-ATLA-JACK-I2-05128

Table 1: URN name examples for the proposal

By using urn in this manner we can have a standardized format for identifiers. Further it allows for easy parsing of the name of the resource and the object authority(ies). It also allow the encoding of authorities using characters in the valid IDN character set.

Conclusion

We have proposed abandoning GGID's as identifiers in GENI. We have proposed using URN and email addresses as identifiers for GENI. We have shown how this identifiers can be written and the benefits of using a human readable namespace for identifiers. We have also proposed a standarized name convention for GENI identifiers.

Bibliography

1. Larry Peterson (Editor), Slice Facilities Architecture v 1.10. http://groups.geni.net/geni/attachment/wiki/GeniControlBr/v1.10%20%20080808%20%20sfa.pdf. 2008.
2. GENI Facility Design. http://www.geni.net/GDD/GDD-07-44.pdf .March, 2007
3. Ted Faber. GENI Identity Document. http://groups.geni.net/geni/wiki/GeniControlIdentity . January 2008.
4. R. Moats. RFC 2141: URN Syntax. http://www.faqs.org/rfcs/rfc2141.html May 1997.
5. Norman Walsh, John Cowan and Paul Grosso. RFC 3151: A URN Namespace for Public Identifiers. http://www.ietf.org/rfc/rfc3151.txt . August 2001.
6. IANA. Uniform Resource Names Namespaces. http://www.iana.org/assignments/urn-namespaces/ . December 2008.
7. ITU Recommendation X.667. http://www.itu.int/ITU-T/studygroups/com17/oid/X.667-E.pdf . September 2004.
8. Paul J. Leach and Michael Mealling and Rich Salz. RFC 4122. A Universally Unique Identifier (UUID) URN Namespace. http://www.faqs.org/rfcs/rfc4122.html . July 2005.
9. David Cooper, Stefan Santesson, Stephen Farrel, Sharon Boeyen, Rusell Housley and Tim Polk. RFC 5280. Internet X.509 Public Key Infrastructure Certificate. http://www.faqs.org/rfcs/rfc5280.html May 2008.

Appendix: Proposed URN representation of Internet2 and National Lambda Rail (NLR) resources.

While Geant2 has a URN namespace assigned to them, neither Internet2 nor the NLR such namespace assignment. Therefore is it necessary to use the publicid namespace to refer to objects in these two namespaces in a uniform way.

The following string representation of backbone resources is suggested:

```
"IDN $BACKBONE-IDN $RESOURCETYPE $RESOURCENAME"
```

which translates into the publicid name-space as:

```
"urn:publicid:IDN+\$BACKBONE-IDN\+$RESOURCETYPE\+\$RESOURCENAME"
```

For example the internet2 circuit I2-ATLA-JACK-I2-05128 would be transformed as string as:

```
"IDN internet2.edu//circuit I2-ATLA-JACK-I2-05128"
```

which translates into the publicid name-space as:

```
"urn:publicid:IDN+internet2.edu:circuit+I2-ATLA-JACK-I2-05128"
```