Access Control for Federation of Emulab-based Network Testbeds*

Ted Faber
USC/ISI
faber@isi.edu

John Wroclawski
USC/ISI
jtw@isi.edu

Abstract

This paper describes a resource access control system for federation of Emulab-based testbeds within the DETER federation architecture. The system is based on three levels of principals and uses generalizations of the Emulab project system to assign access rights. A prototype implementation is described.

Introduction

This paper lays out a model for granting experimenters controlled access to multiple Emulab [White02]-based testbeds in order to establish federated experiments, and describes a prototype. The model generalizes the single-emulab resource access mechanisms to a federated environment. Access decisions are based on the identity of the requesting experimenter as well as projects or testbeds associated with that experimenter. The prototype uses several extensible technologies and is in use federating experiments.

Network testbeds are invaluable for modern research, making experiments more realistic and reliable. They can be used to confirm the dynamics of simulations of networks and distributed systems, to evaluate the behavior of existing network artifacts (viruses, worms) under controlled conditions, and to examine the interactions between a proposed system and existing infrastructure. Doing this work on physical hardware in a laboratory environment to which others have access improves the quality of research.

Federation – combining the resources of more than one independently controlled testbed – enhances the utility of testbeds significantly. First, experimenters can simply access more resources, increasing the scale of their experimentation. Furthermore, individual testbeds may include unique hardware or configuration properties that allow experimenters to embark on new kinds of experiments. Finally, because testbeds act as gathering points for experimenters in a given field, combining testbed resources can promote collaboration between those groups. Security experts and malware architects can test each other's work in a testbed built partially from each group's home testbed. Such collaboration can be cooperative or competitive.

Federation resource management has two complimentary requirements: testbeds make resources available to experimenters without abdicating control. Resources are subject to the policies and constraints of the testbed that shares them. It is this second constraint that both makes federation difficult and provides its power. Our access control enables this controlled sharing.

Federation Basics

This section describes the basic model and workflow of our federation architecture. An experimenter creates an experiment using whatever domain-specific experiment creation tools are available. Should the experimenter or the tools decide that the experiment needs more resources than one testbed can provide or that properties generated by federation are key to the outcome, the tools will invoke the federation system. After evaluating experimenter requirements and available resources, the federation system will divide the experiment among testbeds subject to His or her constraints, create sub-experiments on each and then interconnect them to form the federated experiment. The instantiation and interconnection will be transparent to the user, unless there is a reason to expose it.

After the federation tools have split the experiment, the first phase of embedding the sub-experiments is gaining access to remote resources. Though testbeds have made a decision to allow their resources to be shared, individual access control decisions are made for each experiment. The model and mechanism for this access control negotiation is the subject of this work.

Access control to federated testbeds is difficult because of the very loose affiliation of the testbeds and the expectation of very large scale. Because testbeds are individually managed and their independence generates their unique properties, the factors considered in their access control decisions may be similarly independent. Two testbeds whose resources are being shared in the same experiment may have very different access control criteria.

Federating a few large testbeds is an easy way to create large

*This material is based upon work supported by the Department of Homeland Security, and Space and Naval Warfare Systems Center, San Diego, under Contract No. N66001-07-C-2001. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Homeland Security for the Space and Naval Warfare Systems Center, San Diego.

experiments, but the choice of federating many smaller testbeds is also attractive. A federation system that allows many 10-15 computer testbeds to come together may accumulate more resources than one that supports only a few large testbeds. Avoiding centralization and global naming or trust authorities removes key scaling barriers.

Any federation access control mechanism must address three areas. It must establish a namespace for requesters and providers of service so that they can communicate and reason about access control decisions. It must define a model for testbeds to control access to their resources, including the types and granularity of access controls. It must define an access protocol and interchange format that is clear and extensible.

We have developed an approach for federation within Emulab-based testbeds, called herein the DETER Federation Architecture. This paper describes the resource access control aspect of that architecture. Our design is based on generalizing the controls in a single Emulab while supporting scale and customization. The system defines principals generalized from the Emulab notions of experiment, user, project, and testbed. Principals can join the federation universe without consulting or coordinating with a central authority.

Granularity of access control is based on Emulab's mechanism of granting access to testbed resources through projects. Within a single Emulab, a federated experiment is represented as a project that has appropriate access permissions. Such projects may be created on the fly in response to federation requests, or a static collection of access classes may be maintained. When allocating an experiment to an access project, the hosting testbed bases its decision on the user, project, and testbed names (credentials) that the experimenter asserts. This allows the same experimenter to acquire different access levels depending on the credentials presented.

The access request protocol is expressed in the Web Service Description Language (WSDL)[wsdl] and is extensible at several points. Transport Layer Security (TLS)[rfc4346] is used for mutual authentication and encryption. Standard tools can generate code to make and serve requests across this interface using a variety of implementation languages.

In the remainder of the paper we briefly describe the DETER federation architecture, then discuss in more detail the access control model and how it fits in that architecture, together with our prototype implementation, **fedd**.

The DETER Federation Architecture

The DETER project has created a federation architecture that frames the various components needed to interconnect testbeds. The architecture is designed to scale to hundreds of testbeds and thousands of machines. The immediate goal of the architecture is to guide the interconnection of Emulab-based testbeds.

The full federation architecture must meet three goals. First, it must provide experimenters and their tools with sufficient information to guide the process of decomposing experiments into testbeds. To accomplish this the architecture must provide scalable channels for testbeds to advertise or respond to queries about the resources they permit to be federated; this information may be filtered based on the identity of the experimenter or abstracted for scaling. Secondly, experiments must be decomposed and embedded into federated testbeds – we focus on this below. Finally the architecture must support experimentation across the federated experiment. Part of this goal is to generate a cohesive, scalable experimental environment that may be represented differently to different experimenters. For example, experimenters representing attackers and defenders in a competitive experiment may be provided limited knowledge of their opponents' topology. This paper focuses on the decomposition and embedding facets of the architecture – specifically the access control.

The experiment decomposition and embedding phase of the DETER federation architecture can be viewed from several perspectives – experimenters, the federation system, and the federants all see the architecture differently. We discussed the experimenters' view of federation in the introduction, and focus on the system and federants viewpoints here.

For the system implementers the centerpiece of the federation system is the *federator*. It takes input from experimenters or their tools and creates an experimental environment split across federant testbeds. Specifically, the federator decomposes an experimenter's annotated topology into federable sub-experiments, acquires access to appropriate federants, embeds the sub-experiments in federants, and then connects them into a shared environment. Figure 1 illustrates this architecture.

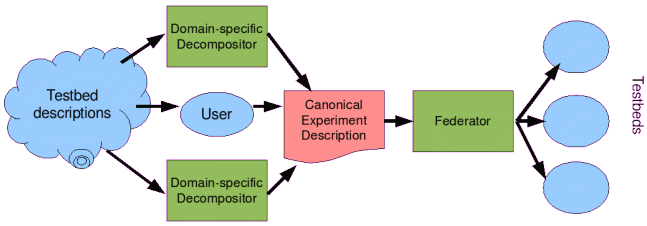

Figure 1: DETER Federation Architecture

The architecture is partitioned to separate concerns of the various players. The partitioning of the experiment into pieces suitable to federation depends on the nature of the experiment. This split must be guided by the experimenter using knowledge of the resources provided by the federation system. For example, an experiment used to study throughput of a new protocol must be aware which links are inside a testbed and completely controlled and which are not, to ensure that the unpredictable link performance does not invalidate the results. Collaborative or adversarial experiments will divide along the lines of visibility and testbed administrative boundaries.

The output of this splitting step is an annotated topology description in a standard language, annotated to facilitate the

decomposition. The federator accepts these experiment topology descriptions. Currently this language is the Emulab topology description language, based on the ns simulator language. Each node is annotated to indicate the testbed in which it should be embedded. This is a standard but low-level format: we assume that in most cases this description will be generated by higher-level, more sophisticated tools. The division allows development of domain-specific annotation tools to proceed at the same time as the federator is advanced.

On the other end, the federator must communicate with federant testbeds for two basic operations: requesting resource allocation within the federant and embedding a topology subgraph on that federant. Emulab-based testbeds have interfaces for embedding topologies remotely, and the DETER architecture uses those interfaces directly. Resource allocation and management is provided by our software.

Once the embedded experiments have been formed into a cohesive environment, the federator makes the environment available to experimenters. The federator may present different views of the environment to different experimenters as we described above, though the details are beyond the scope of this paper.

We are in the process of instantiating this architecture on DETER and other testbeds. Experiment creation is operational, and higher level functions are being developed.

Access Control Model

The federation access control system is faced with three design problems: the granularity and mechanism of granting access, the naming and validation of principals, and the protocol for requesting and granting access. In addressing each of these problems we followed three tenets: that the system should minimize scaling bottlenecks, that the independence of federants be preserved, and that the system be applicable to federating Emulab-based testbeds.

Emulab-based testbeds control resource access primarily through the collection of projects established on the testbed. Each user is a member of one or more projects and each experiment is associated with a project when it is created. When a user creates an experiment, the rights to access restricted resources are based on the project rights. Users can only instantiate experiments from projects to which they belong. (Users also authenticate themselves to gain access to the testbed as a whole and vary in their ability to create experiments, but the project system controls resource access.)

We extend this model directly. When the federator requests access to an Emulab-based federant, it is granted the right to instantiate an experiment under a project in the federant. The federant may instantiate a new project to house the federated experiment or it may provide access to an existing project with an appropriate set of rights. Creating a new project for each experiment provides the finest control over the access granted; the new project has exactly the rights requested. A testbed operator may also choose to select classes of access that the testbed will allow and create static projects with those permissions, assigning individual access requests to the appropriate project. Dynamic projects are more expensive at access time, static projects are more coarse. Neither choice requires operator intervention when access is granted. Dynamic projects are created automatically within the bounds set by the operator.

Directly extending the project-based access control minimizes the changes needed for a testbed to join the pool of available federants, however resource management capabilities are limited to the node-based Emulab controls. Some testbed resources are difficult to control with the existing project-based access control – for example switch capacity. However, the federant retains the same control over resources that it exerted in an unfederated world, using the same mechanisms.

Principals

Single-site Emulabs characterize users both by their username and by their project memberships, though the project membership is the primary access control mechanism. In order to be more flexible in controlling remote access, we export users, projects, and testbeds into a global set of principals. These principals are expressed in a global format, but are not allocated from a central system.

We define three kinds of principals: users, projects, and testbeds. Access may be granted to any of these principals, based in part on their kind. In a federated environment, users are a more prominent feature than within a single testbed and we recognize this by making them principals.

Aside from allowing access rights to be conferred directly on users, user principals allow the system to express anonymous users or users unaffiliated with any testbed. Testbeds as principals allow us to express both the interests of federable testbeds as a whole and to express the connections between users and testbeds. “Peering Agreements” between testbeds are a common scalable way of allocating access; testbed principals reflect this directly. Finally, because projects are the basis of current Emulab node access control, we also expose them as principals. It is convenient to be able to allocate access to groups working on the same experiments, and projects allow us to express this concept.

Our decision to add users and testbeds to the project-driven access control broadens the domain of discourse with respect to resource access in a federated world. Having defined the kinds of principals, we define a name space to allow testbeds to collect and share data about them.

Naming Principals

An effective system of naming principals for federation must support several properties to promote scalability. Some form of shared naming semantics is convenient so that testbeds can agree on principal identities and reason about their rights.

In particular, we advocate simple global names that:

- Can be authenticated without requiring a third party
- Impose minimal constraints on additional access

control information

- Can be created without recourse to a third party

The first of these allows the system to operate without the performance or scalability bottleneck of a central authentication point. More to the point, each authentication becomes a local discussion between the parties involved, reducing dependencies and complexity. Finally, it avoids the thorny problem of authenticating names between parts of the system that do not agree on a third party they trust for authentication.

Testbeds will independently determine and manage their access strategies. Though our system of principals encourages access controls that similar to current strategies, that structure is not required. The urge to tie naming and access rights is strong and we specifically avoid it for flexibility and growth.

The argument for being able to create names without third party entanglement includes the arguments for independent validation, but more importantly enables anonymous names. If some third party must create a name, it creates a potential privacy leak. Autonomous naming enables anonymity. We believe that autonomous names can be implemented efficiently.

One form of naming that meets these criteria are self-certifying names as used in the Self-certifying File System [Kaminsky03]. We use a simpler version in fedd.

Proxy Requests and Attestation

Being able to assert an identity is the basis for federated access control decisions. The federator can make access requests directly on behalf of experimenters with their credentials, or it may ask a proxy for the local testbed to attest to credentials for the experimenter.

The federator may initially make a request for resources using the experimenter's global user name. If the remote testbed knows and trusts the user, this may be sufficient. A more likely situation is that the operators of two testbeds have worked out a federation agreement and that they honor requests attested by testbeds.

The federator can contact a local proxy that trusts the user and is empowered to speak as the testbed principal. The proxy can amplify the user's request by including information about the user's testbed or project affiliation and send it to the remote testbed. Trust between the user and testbed allows the user to combine their rights with those of the testbed.

This amplification uses the same access control mechanisms, but creates a new credential. Principals can use this mechanism to cooperate and generate requests that combine the rights of both parties. Credential amplification uses of existing code and Emulab configuration for easy deployment on existing testbeds.

Request Protocol

When requesting resources a federator, acting on behalf of some principal, creates a set of assertions about the request and sends it to the potential federant. This exchange includes a mutual authentication of the principal names so both are certain of the identity of the other end. The assertions include information about the resources to be requested, the access time and duration, additional principal information (e.g., project and user information from a testbed requester), and a mechanism to be used for access. The federant evaluates the request in light of the local information it has about the principal. For example, a user may not be allowed to make assertions about testbed or project affiliations. Principals that the testbed has no previous arrangement with may be denied access entirely. No explicit principal type information is passed, so a principal may be treated as a testbed by one testbed and as a user by another. Table 1 lists request fields.

Field	Purpose
Testbed	Originating testbed
Project	Originating project
User	Originating user
Allocation ID	Name of the federated experiment
Resources	Characterization of nodes and capacity to be requested
Access info	Authentication information for access, e.g., a public key
Timing info	Predicted embedding time and duration

Though resource and access time information is included, this is an access request. A successful reply from the testbed indicates that the federator has permission to acquire the resources from this federant, not that the resources will be available at the request time. The protocol acquires permission to embed an experiment, it does not allocate resources.

Assuming that the requested access is granted, the federant will establish or arrange access to a project with the relevant rights and return the information necessary to access the project. This includes the local project and user names to use as well as the internal names of the various Emulab service sites within the local testbed. These services are necessary to establish the federated environment.

Simplicity has led us to a request/response protocol rather than a complex negotiation. A federator may have to make several requests using different user, project, or testbed credentials until it finds the set that a remote federant will accept. Local data can be used to guide this search.

The simple request/response can be used to build up more complex negotiation strategies, but defining a complex negotiation protocol at this stage may lock us into a wrong paradigm.

Model Summary

The model we described provides a request/response

mechanism for an experimenter to acquire access to an Emulab-based testbed to create a federated experiment on it. The experimenter has self-certifying names that characterize it to a remote testbed as a user, member of a project, and/or as a user of a particular testbed. A request is made by one of those principals, perhaps asserting additional identity information, to the candidate testbed. Based on local information about the requesting principal an the content of the request, the candidate either grants the request using the local Emulab's project system to grant resource access, or rejects the request.

We have implemented a prototype of this system, which we describe below.

Fedd: a Federation Access Control Daemon

Fedd instantiates the architecture above including the global names and multi-level access controls. It runs on the control or *boss* node of an Emulab testbed, responding to access requests and creating the necessary dynamic projects or configuring existing ones for remote access.

It provides a Web Services Description Language (WSDL) specification of the access request interface with integrated TLS encryption and authentication. Dynamic project creation is supported. We describe the implementation in detail below.

X.509 Certificates and Global Names

One implementation for global names is to tie the identity of a principal to the possession of a public key. The public key becomes the principal's name. A principal can prove their identity by responding to a challenge encrypted in the public key. If keysize is sufficiently large it is extremely unlikely that two principals will generate the same public key.

The current implementation of fedd uses a hash of the public key as the *federated identity* (fedID) and communicates identities using X.509 certificates [x509] [rfc3280]. Fedd ignores the principal and issuer names and uses the certificates as a mechanism to pass keys. Existing TLS code authenticates federators and federants.

By using the hash of the public key as the fedID, fedd gets a uniform representation of identity while allowing principals to create whatever format key is acceptable to their security concerns and sensibilities. There are standard TLS routines to hash public keys and therefore create federation identifiers.

The simplest way for a testbed operator to list the federators or testbeds with which it will communicate is to put self-signed certificates from each of them into the file fedd uses to authenticate certificates. This is the equivalent of listing their federated identifiers in a flat file. In addition to the certificate file, each fedID has an assigned type with respect to this fedd.

Fedd uses X.509's certificate chains to allow introductions. A principal that presents a certificate and responds correctly to a challenge is accepted as a user principal even if fedd has no certificate for it, if the certificate presented is signed by a testbed that fedd trusts. This allows the testbed to introduce remote testbeds to users. This introduction capability can be restricted to certain principals or disabled altogether.

Access Control Specifications

Fedd allows testbed administrators to describe their access control decisions in terms of the three-layer principal model above. A triple of (testbed, project, user) in the federated name space is mapped into a (project, user) pair in the local name space. Any component of the triple can be wildcarded to match any incoming value. The components can also specifically be marked to only match empty values. This allows one to specify that a given user is mapped one way when presented by any testbed in any project (<any>, <any>, user) and another way when acting on their personal credentials (<none>, <none>, user).

All the names in the triple can be given as federated identifiers, but for clarity it is sometimes helpful to use local human-readable names. This also allows testbeds to expose parts of their internal name spaces to one another without generating and exposing fedIDs for the internal names. An example is a testbed that attests that a request is made on behalf of a user in the “emulab-ops” project on its local testbed. This project is present on all Emulab-based testbeds and is a convenient conventional marker for testbed staff.

The local pair may indicate an existing project to which matching requesters are granted access, or that a project is to be allocated dynamically. Either the user or the project may be marked as <dynamic> meaning that the relevant entity will be created. The current implementation only supports the creation of dynamic users in dynamic projects, but the more complex functions are being added.

The pair of local user and local project is also annotated with the node access abilities of that project. In the case of a request mapping to an existing project, the annotation describes the existing access capabilities of that project; in the case of a dynamic project, the annotation represents the access capabilities granted to the new project. In either case, fedd is able to compare the access power of the local project with the resources requested in the access request and deny requests that would access forbidden resources.

Below is an example access specification that maps users from the local emulab-ops group on a requesting testbed to a user with the same name in the DETER1 project on the local testbed and that maps a user identified by a fedID to the DETER project, faber user on the local testbed. The DETER project has access to nodes of type pc3000, which the DETER1 project does not.

(<any>, emulab-ops, <any>) -> (DETER1, <same>)

(<none>, <none>, fedid:12ecc7415746281efa0ed58e180c51a5cba13a57) -> (DETER:pc3000, faber)

Request Protocol Details and Specification

Fedd's request/response protocol is specified in a WSDL document at http://www.isi.edu/~faber/fedd/fedd.wsdl and

contains the fleshed out details of the protocol described above.

The contents of the request are as described above. Alternative representations for most names are supported, as are several varieties of key formats for access information. As a practical matter, the current implementation uses fedIDs and SSH keys for the names and access keys, but there is room for expansion.

In the request, resources are characterized in Emulab terms. Node counts are given for each type/image pair of nodes. This section is used by fedd to confirm both that the testbed supports these node types and images and that the user making the request can access them. Additionally the network capacity required by the experiment is given as either a peak or average value. No attempt is made to fully characterize topology here in the access phase.

On return, access information is annotated with attributes that the federator can use when establishing the shared environment.

For testbeds that do not dynamically instantiate projects for access control, fedd can run remotely, and the DETER project uses it that way on occasion. In general, fedd should run on the *boss* node of the controlled testbed.

Proxy Requests

As part of their internal operation, Emulab-based testbeds assign each user an X.509 certificate signed by a certificate authority based at the testbed. Our system handles proxy requests by running a per-testbed instance of fedd that is willing to attest to valid user requests using testbed credentials for users holding a valid local testbed certificate. Of course, when the user makes claims to be part of a given project, the fedd validates such assertions before forwarding them.

In cases where project credentials have meaning beyond a single testbed, a natural implementation is to run separate fedds on their behalf, unless the project members were willing to accept credential replication. If the project exists on a single testbed, that testbed's fedd can simply provide the project credential instead of a testbed credential, should the user request that.

Fedd is an operational federated access control daemon built on existing technologies and integrated with current emulab configurations. It is in current use creating federated experiments across DETER and other testbeds.

Conclusions & Future Work

This paper has described a federation access control system for federating Emulab-based testbeds. Individual testbeds control access to resources through the existing Emulab projects system, dynamically adding projects when it is convenient to do so. System principals are testbeds, projects, and users which are generalizations of the Emulab constructs of the same name. The principal identifiers are drawn from a self-certifying name space for scalability and flexibility.

We have implemented a prototype of this system based on existing technology and compatible with Emulab configuration conventions. That system has proved useful in practice in creating federated experiments.

Though the access model ties to Emulab by adopting the three-level principal system and in expressing allocations in Emulab terms, these are customizations rather than central features. The three levels of aggregation reflect generic testbed organization and are applicable in non-Emulab environments. When a testbed grants an allocation, the current protocol describes that allocation in Emulab terms, but that response description is typed and isolated in the message. The protocol is designed to support other allocation descriptions.

As the DETER federation architecture in general expands to interoperate with more kinds of federants, the access control system will also be extended. DETER is investigating federations with provisioned access interconnection networks as well as traditional testbeds.

We believe that the fundamentals of the access control system are sound and extensible enough to operate in a more general federation environment than inter-Emulab federation. Extending the protocols to support more general resource descriptions and the model to accept other principals are natural ways to extend into those domains.

References

[Benzel06] *Experience with DETER: A Testbed for Security Research*, Terry Benzel, Robert Braden, Dongho Kim, Clifford Neuman, Anthony Joseph, Keith Sklower, Ron Ostrenga and Stephen Schwab. In *Proceedings of Tridentcom (International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities)*, March 2006.

[Kaminsky03]*Decentralized user authentication in a global file* system*, Michael Kaminsky, George Savvides, David Mazières, and M. Frans Kaashoek. *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, October 2003.

[rfc3280] *Internet X.509 Public Key Infrastructure Certificate and Certificate revocation List (CRL) Profile*, Russel Housley, Warwick Ford, Tim Polk, David Solo, *RFC 3280*, ISOC, April 2002.

[rfc4346] *The Transport Layer Security (TLS) Protocol Version 1.1*, T.Dierks, E. Rescorla, RFC4346, ISOC, April 2006.

[White02] *An Integrated Experimental Environment for Distributed Systems and Networks*, by White, Lepreau, Stoller, Ricci, Guruprasad, Newbold, Hibler, Barb, and Joglekar, appeared at OSDI 2002, December 2002.

[wail] http://wail.cs.wisc.edu/projects.html

[wsdl] *Web Services Description Language (WSDL) 1.1*, Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, http://www.w3.org/TR/wsdl

[x509] *ITU-T Rec. X.509: Information Technology Open Systems Interconnection – The Directory: Interconnection framework*, June 1997.