A Path to Evolve to Federation of TestBeds

Soner Sevinc

Princeton University

Abstract. Federating the numerous existing networking testbeds offers multiple possible benefits, but so far testbeds and federation efforts remained semi-independent, and not wholly united as a single federation. We identify several problems against federation, namely, establishing new trust relations and agreeing on a common schema between the entities. In this paper we explore practical ways to set up federations by making use of the existing organization to user, and inter-organizational trust relations. First we give an analysis of the existing testbeds and federations in terms of their trust models, and the set of technologies they use. Next, we argue for a design of a federation which leverages existing trust relationships. Our prototype implementation shows how readily this design can be implemented using a minimal set of schema and technologies being used by the existing testbeds and federations. Using our analysis we then show how this design can be expanded for larger scale federations, and give examples of possible future trust models.

1 Introduction

Federation of testbeds is a key step to set up a ubiquitous computational networked experimentation infrastructure. While a single testbed has limited computational resources in quantity and variety, multiple federating testbeds can provide the user with a wider set of resources for network and systems experimentations. Contemporary examples are seen in PlanetLab federation with Europe and Japan partners, also GENI [6] federation, where there is an ongoing effort to allow users from different testbeds be able to use resources from all testbeds, without requiring users to sign up to each one of them separately. So far testbeds and federation efforts, although being somewhat related, have not merged into a single federation of testbeds.

Migration from a set of monolithic testbeds or smaller federations into larger federation of testbeds faces major challenges. These challenges consist of cross-testbed user authentication, ease of entry/use for users, and also human factors such as different testbed operators coming to an agreement on a set of federation compatible technologies, data formats and security standards.

User authentication is a more complicated problem for federations compared to a single testbed, since users are dispersed among numerous domains rather than one. As an example, today a user who wants to use computational resources at Emulab needs to contact operators out of band, like email, phone, etc, and they decide whether to grant access or not. Generally users are granted resources if they are known by operators in person, or by checking the email domain of the

T. Korakis et al. (Eds.): TridentCom 2011, LNICST 90, pp. 126–141, 2012.
© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2012

users to ensure they are from a trusted university or institution. Such today's ad-hoc methods are inefficient, and can be less reliable since human factors opt into the decision process. This is especially important when number of candidate users of testbeds grow with federation, hence it becomes harder to set up trust relations in ad-hoc ways. Since there is generally no direct knowledge between every testbed and user in a federation, indirect trust relations should facilitate authentication in federations.

Even the trust is logically established, that is, the user is authenticated, it is not much meaningful if it is hard for users to use the system. Therefore, ease of use is crucial for a working trust relationship between testbeds and the users. As an example, one of the major concerns faced in pulling more users to a computational facility is seen in the grid community [19] where authentication of users based on public cryptography is observed to make it harder to join for users rather than using user name and password based authentication. This is not only because the users find it harder to manage public and private keys, but also it is possible to utilize the user name and password a user already has, such as from a university, so that a separate joining overhead for the user is bypassed. It is common sense that users will always choose simpler to use testbeds rather than with high barrier to entry, which is important especially in a federation where users are served multiple choices for networked resources among which they will allocate from to run experiments.

Trust establishment between testbeds or other organizations depends on a mutual benefit principle. Example of such a trust can be an agreement between PlanetLab and Emulab to share each other's resources. It is important for two organizations to fully understand each other's trust structure, including user base, and other partners before establishing authentication or authorizations trust relationships. This requires an analysis methodology which reveals the trust models used by the organizations.

Adopting a common federation schema is one technical barrier into setting inter-testbed trust relations. Such a schema consists of the common standards for security token formats, the protocols to exchange tokens, and also resource specifications(RSPEC) and API operations. Almost every testbed grows in its own community, developed by a group of researchers and acquires a particular user base. When it comes to federate two testbeds, one method is to make them talk compatible protocols and develop a conversion layer in between which translates the security tokens, API operations, and RSPECs. Nevertheless, the conversion layer method is insufficient for larger scale federations, since it is generally a huge task to make all standards and operations compatible with each other, and it is impractical to develop a layer between every two testbeds in a federation. Therefore, another solution is having all testbeds adopting a small set of common standards, but there are human factors that prevent this. Testbeds tend to grow vertically; serving their own community first, acquiring more users and getting more specific in their own standards and operations, rather than horizontally; being more compatible with other testbeds. Therefore, as time goes by it becomes more difficult to even create interoperability layers between two testbeds.

In this paper we describe not only a federation architecture, but also a path to practically realize it. We analyze the existing trust relations of current testbeds and federations, and then leveraging them in order to establish user to organization and inter-organization trust relations at a large scale federation. Our implementation of this design also addresses the problems of ease of use/entry, and common schema by making use of the best and minimal practices among available technologies. While trust relations analysis guides how inter-connections between different federations should be made, the implementation explicitly shows the inter-connections between software components of different systems.

Section 2 will give an overview of the current testbeds, federation and technologies they use. Section 3 will first give an introduction to trust models, a formalism which we use to analyze federations. In Section 4 we analyze the existing systems and see how they are related to each other and fit into the bigger picture, by applying the trust models formalism of Section 3. In Section 5 we show both the design and our implementation of this design for GENI federation, which realizes a first step of a large scale federation. Using our analysis, we also argue about how future evolution of the federation can occur and give an example trust model for the future.

2 TestBeds and Federations

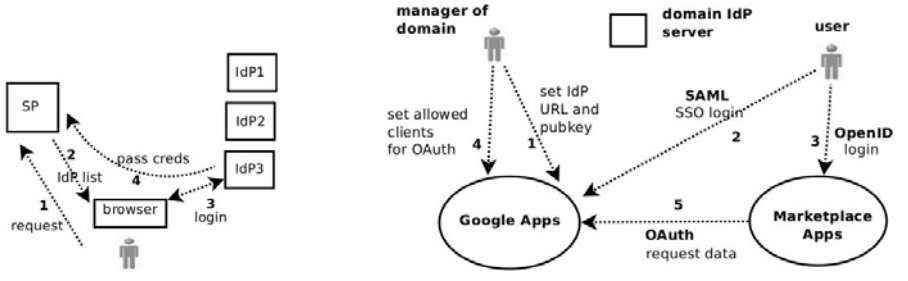
In this section we explain some of today's testbeds, federations, and technologies they use and give their relevant properties that will be of use to our analysis in Section 4. These testbeds are candidate members of a future large scale federation. It is important to investigate the current technologies used and practices held in testbeds and federations in order to select the best among them for a future large scale federation.

PlanetLab/SFA/ProtoGENI/GENI. PlanetLab is a widely used network and distributed systems research testbed, which has been been centrally managed by Princeton University. Slice-based Federation Architecture(SFA) [14] is an effort again led by Princeton that also spans PL-like facilities such as PlanetLab Europe, Japan, Vini and MAX, aiming to federate these testbeds. SFA architecture is also in parallel with other federation efforts, like ProtoGENI, which is Emulab's federation effort [15]. As a result, PlanetLab and ProtoGENI have different federation implementations of the same architecture. Besides those testbeds, there is the GENI effort, led by the GENI Project Office (GPO), to promote and coordinate sharing of network research tools and resources, and the compatibility of testbeds, starting with an effort to ensure that APIs and credential formats are standardized between testbeds.

We give some terminology commonly used by all four. A *component* is representation of a testbed resource, such as a single machine, programmable router, etc. An *aggregate* is a collection of components, managed by an *aggregate manager(AM)* at a particular testbed. Users are allocated a *slice* of overall resources a testbed manages, made of individual virtualized resources from many networked

components or aggregates, where individual resource units are called *slivers*. An *authority* manages the allocation of slices to users at each testbed.

SFA federation uses a certificate based scheme in expression of user identities and slices, and a simple logic governing authorizations for actions in the system. For example PlanetLab users have to acquire a "slice" credential in the form of a certificate, from PlanetLab, which they can later use as a parameter to API operations for allocating resources on a federation partner aggregate such as Vini. Similarly, "user" and "authority" credentials are used for user related and administrative operations.



(a) SAML Single Sign On (b) Google Apps use of IM systems

Fig. 1. Federation technologies and their use

InCommon/SAML/Shibboleth. InCommon [7] is a federation effort that aims to bring together universities and research labs from US for sharing of university resources, such as academic documents. For example, if Princeton wants to share some material, it creates an InCommon login web page, then students from Cornell University can log in to that web page using their Cornell University user name and password, and access the documents shared by Princeton. While InCommon is not directly related to testbeds, it has an important feature such that it serves as a rich target user base for most testbeds, with 247 participant organizations which contain the people from the academic community.

As opposed to SFA, InCommon users do not use certificates, but a federated identity management(FIM) standard called SAML [17] for federated authentications and authorizations. The code used by InCommon that implements this standard in is Shibboleth [18], which is open source. Every member organization runs Shibboleth software which allows users to access resources at different domain with single sign on, where all user related information such as affiliation, email, etc are transferred by Shibboleth from the user's home domain to the target domain, determining if user can access or not, or to what degree. In this scheme, users do not deal with public and private keys, or obtaining credentials manually in the form of certificates. The encrypted passing of user credentials are handled by the FIM system. Shibboleth and SAML are the only FIM solutions and standards supported by InCommon support team and the federation.

Figure 1(a) exemplifies the SAML SSO protocol. In the first step, a user visits the web site of a service provider(SP), from which he/she wants to use some resource. Second, the user is returned a list of known and trusted identity providers(IdP) by SP among which he/she can select his/her home organization if exists. If exists, in the third step, the user is directed to it, in this case to IdP3, and then authenticated, and fourth, directed back with credentials to SP, at which he/she can access service if authorized.

Google Apps and FIM. Google Apps is an environment where both Google and independent organizations inter-play for request and servicing of applications. More specifically, in the Google Apps world there are two kinds of applications; the default "Google Apps" provided by Google such as mail, calendar, contacts and documents, and also the "marketplace applications" or "installable Google Apps" which are third party developed applications. Users of Google Apps are generally associated with a domain called a partner organization. A domain manager can choose to install from the marketplace apps, by searching and selecting from a list, so that its users can use them.

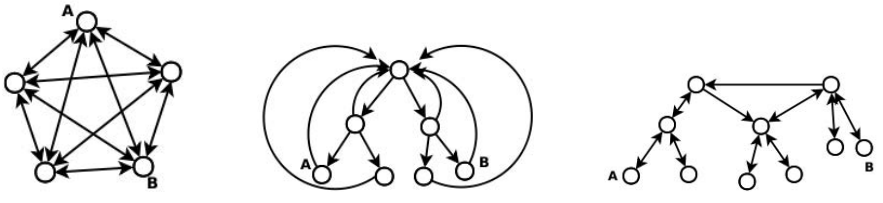
Google Apps makes use of the federated identity management technologies SAML, OpenID and OAuth. OpenID is another standard like SAML for FIM and mostly used for authentications, such as, a user can log in to OpenID supported web sites using their Google or Yahoo IDs. OpenID runs a message exchange protocol between domains to authenticate the user. OAuth is another FIM solution which focuses more on resource federation on web, such as documents, photos, etc of a user residing in one domain can be transferred and used by another domain on consent of user, and in a secure way.

Figure 1(b) shows how Google Apps makes use of these three technologies. In step 1, the manager can make configuration at Google so that user authentication and authorizations are handled by the SAML IdP server of the domain. By this way, in step 2, the user can log in to Google Apps with SAML. Next, every logged in user can also seamlessly log in to marketplace apps using OpenID at step 3. Each such application has to implement OpenID login, and they receive the ID of the user from Google, without need to authenticate users by themselves. At step 4, again the manager can make configuration at Google such that some marketplace apps are allowed to use user resources such as calendar or feed data. Then these apps use OAuth technology to get and use those resources as a part of what service they are providing at step 5.

ABAC. While FIM systems is one candidate family of technologies to realize federated authentications and authorizations, another is the distributed trust management systems, which are certificate based. Attribute based access control(ABAC) [9] is one trust management system, which was proposed as a candidate for GENI authentication and authorizations. Different from SFA, ABAC like systems, e.g. SD3 [8], have a strong logic piece which can do formal inferences to rigorously perform and audit access control decisions. The ABAC policies are expressed in terms of rules and facts of a formal logic. We will further discuss ABAC in our analysis at Section 4.

3 Trust Models

A trust model shows the trust relationships in a distributed system. Examples of trust relationships are such that, PlanetLab can trust Stanford University IT department in attesting valid students. Similarly ProtoGENI can trust Taiwan University for its students. Additionally, PlanetLab can trust every university IT that ProtoGENI trusts in attesting new students, by this way PlanetLab can get to verify students indirectly through ProtoGENI. Trust relationships similar to this example exist and play an important role in a federation of testbeds. In this section we will explain the trust models formalization and explain some of the models explored in literature, and in the next section we apply the formalisms to various systems and technologies of Section 2.



(a) Circle of trust (b) Hierarchical (c) Distributed

Fig. 2. Trust Models

Figure 2 shows three generic trust models from literature [10], circle of trust, hierarchical and distributed trust models. A solid directed arrow from a node A to node B shows A *trusts* B, which is established out of band between the two entities. More specifically an arrow represents an *assertion* or a policy statement made by A about B, that could be stating a vouching relationship, delegation, etc depending on the policy. An assertion can be implemented as a single certificate signed by A. A *trust path* is a set of arrows that begin with some node A and ends at a node B, showing an indirect trust from A to B.

In the *circle of trust* model (2(a)), there is a direct trust between every entity in the system. The length of any trust path is 1. The downside of this model is that it is difficult for a new node to join the system, since there needs to be $O(N)$ trust relationships be established out of band, for N number of nodes.

In the *hierarchical* trust model (2(b)), every node trusts its children, if any, and the root. One example is from the PKI system. In the process of authenticating a domain like "princeton.edu", there are three assertions required. First, everybody trusts root certificate authority, keeping locally its public key. This trust is depicted with the first arrow going from node A to root node. Next, root has to sign a certificate for "edu" domain, shown by the arrow from root to second level node, and finally third arrow shows "edu" trusts "princeton". It is easier for a new node to join in the hierarchical trust model, which is done just by the new node downloading the root public key. The disadvantage is that the root becomes the single point of compromise that can affect the whole system.

Finally, in the *distributed* trust model (2(c)), there are multiple trust anchors and trust between them. This model fits to environments with more autonomy, where there is not a single root of trust but multiple root like entities. One feature of this model is that trust paths can be longer; as seen in the figure the maximum path length is 5, compared to lengths 1 and 3 in the other two models. The advantage of this model is that the compromise of one or more nodes does not affect the whole system.

4 Trust Model Analysis of Federations

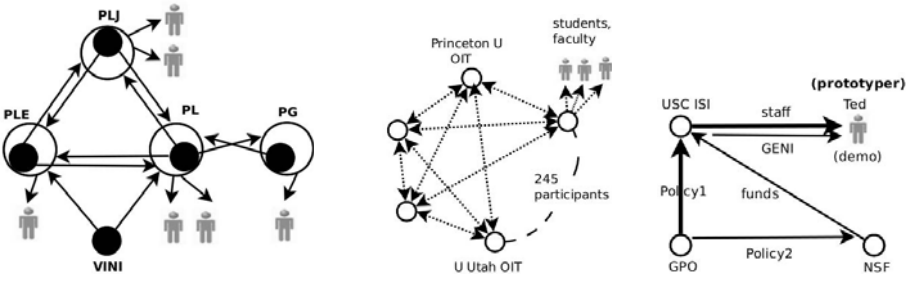
We explore the trust relations in the systems that we talked about in Section 2. Trust relations are sometimes implicit, and it is important to understand those in order to leverage them in a large scale federation. They can shape the overall trust model of the federation, guide the new relations need to be established, and also determine the authentication and authorization mechanisms adopted as we will see in Section 5.

SFA and GENI. The PlanetLab and SFA trust model is shown in Figure 3(a). Each of PL, PG, PLE and PLJ has both resources and users, whereas Vini has just resources. The dark circles show AMs, and the light circles show authorities. For PL* and PG these two are generally intertwined, managed by the same operators, so we draw AM's inside the larger authority circle. On the other hand, VINI is denoted by a dark circle only.

A slice credential is shown by the arrows from an authority to its users. An arrow from an AM to an authority means that the AM trusts the slices from that authority, that is, it can create slivers for those slices. Therefore, a set of arrows beginning at an AM and ending at a user shows that the user can create a sliver at that aggregate, constructing a trust path, as described in Section 3. We assume there are arrows from inner AM's to enclosing authorities, which are not depicted in the figure. In current SFA federation, a slice from any of PL* authority can be used to create slivers at other PL* aggregates, as shown by arrows from AM's to authorities for all PL*. On the other hand, Vini only allows PL and PLE slices; therefore we have arrows from Vini only to PL and PLE authorities. This might be because a need for PLJ did not occur yet or this is not negotiated yet, or because of other such human related reasons.

The interoperability effort between PlanetLab and ProtoGENI, contributed by GENI, is also seen in the figure. While the main focus of this effort is to make the credential format, and API operations compatible, its implication in terms of trust is such that both AM's trust other authorities for allocation of resources. We think such trust relations can be constructed by time between ProtoGENI and other PL's, as well.

The user and authority credentials mentioned before are not depicted in this figure, because the trust relations the user credential entails is very similar to slice credential, and the authority credential is very simple, not having any cross-domain trust implications. In current SFA federation, while the authorities create



(a) SFA Federation (b) InCommon (c) ABAC

Fig. 3. SFA federation, InCommon and example ABAC Trust Models

slices only for their own users, we show in Section 5 how slices can be created to outside users, as well.

InCommon and SAML. The SAML SSO protocol mentioned in Section 2 implies that there has to be trust relations such that, SP trusts the set of IdP's, and IdP's trust their users. The trust between IdP and the user is exposed to the SP at step 4, when user is redirected to SP by home IdP, passing the user credentials with it.

InCommon trust model is shown in Figure 3(b), which uses SAML SSO. For our discussion we can assume that every member organization, such as Princeton University or University of Utah, in InCommon runs a SAML SP and a IdP, although some might choose to just run a IdP if they do not have any resources to share, or just run a SP if they do not have any users. So the figure shows arrows from every SP to every IdP, where we do not depict the SP and IdP pieces explicitly at each node for simplicity. InCommon forms a trust model much like the circle of trust model between the organizations.

ABAC. ABAC is a system that can handle complex trust relations, one example of which is shown in Figure 3(c). The figure depicts trust models of two example ABAC authorization policies. These policies in English are such that GPO says Policy 1:"All ISI staff are GPO prototypers", and Policy 2: "All principals having GENI attribute from NSF funded organizations have 'demo' attribute". Therefore, for a user called Ted to prove that he is a GPO prototyper, he has to collect a certificate saying he is a ISI staff. When combined with Policy 1 statement the proof completes. This is depicted by the trust path drawn by bold arrows, beginning with GPO and ending with Ted. Similarly, to prove that he has 'demo' attribute, first Ted gathers a certificate stating that NSF funds ISI, and also one certificate stating ISI says that Ted has 'GENI' attribute. Combining these two and the Policy 2, proof of Ted having 'demo' attribute completes, which are shown by the light trust path on the figure. What ABAC does in this figure is to discover the trust path, collect assertions, and arrive at a logical

proof out of them to make the access control decision. It is important to note that complex policies can give rise to complex trust models, and hence proofs, which need to be expressed in some formal syntax.

The federation enabler technologies that realize the mentioned trust models are divided into two; the certificate based systems and the FIM systems. The former relies on public cryptography for authenticating users; therefore users need to keep public/private key pairs. One good feature of this is that formal delegation logic based on public cryptography is well explored in literature such as in PolicyMaker [3], SD3 [8] and ABAC. Therefore they can realize complex trust models. Another family of certificate based systems is ones with no formal delegation logic, as in the case of SFA. These realize simpler trust models like circle of trust. The FIM systems, on the other hand, are user name and password based, and the target user base is everyday users rather than advanced users who can manage public/private keys. There is no formal delegation logic in them, and the trust model is generally circle of trust.

The trust model of Google Apps follows from the SAML trust model between user, partner organization and Google which is already mentioned, and also simple trust relations implied by OpenID and OAuth which are not depicted.

5 Design and Implementation

In this section we explain a design, guided by the analysis of Section 4 and its implementation. This design addresses the problem of establishing new trust relationships in a federation, by answering the following specific problems; authentication/authorization, ease of entry/use, and a minimal federation schema, as mentioned in Section 1. We identify both the shorter term and longer term realizable components of the design, and give a prototype implementation of the former. We orient the design and implementation on four principles that we believe are important in federation of testbeds.

(P1) Leverage existing user base Inheriting users from domains with similar user base is one way to address the problem of reaching many users and making it easy for them to join and use the system. As mentioned in Section 1, GridShib is an effort to make the entry to the grid systems easier for users using user name and password logins. In addition, CILogon [4] is a parallel effort which aims to enable users to authenticate over their accounts in their universities, through InCommon federation. This means, all students of universities and members of research labs in InCommon can login CILogon grid computing facilities, as well, just by using their organization user name and password. Another example is seen Google Apps usage of OpenID technology in order to allow third party applications be able to authenticate Google Apps users without need for a separate join phase. Similarly, in social networks, smaller web sites can authenticate millions of users using a larger network's user base. Since the trust between the user and the larger organization was established before, there is no need to re-establish the trust with the user. Our design goal, as well, is to inherit as much of already established trust relations, as possible.

(P2) Minimal set of schema and technologies The large incompatibility between different testbeds and federations favor a common minimal federation schema rather than pair wise compatibility layers. As mentioned in Section 1, the testbeds can differ from each other in terms of credential formats and security protocols among others. Our approach is to identify a minimal and practically realizable set of schema and technologies, avoiding overlapping functionality within it, and keeping it easy enough for every federation participant to adopt. In our design and implementation we primarily focus on credential format and security protocol compatibility.

(P3) Do not over-design/implement the system We choose to primarily implement the system for simpler trust model rather than complex trust model, where the latter can have much more complex system requirements, whose software solutions are not clear yet. Hypothetical use cases entail complex trust models as in Figure 3(c), and require formal trust management systems. But, it is not clear when such use cases will be a real life requirement. We adopt the principle that "make it available immediately, even though no one knows for sure what it will be in the future." [13] which is known by experience that can give rise to a design both viable in medium term, and also the enabler of further developments for the longer term. We believe such an approach will allow an experience driven evolution of the system.

(P4) Two stage design Although we aim to make the system work in shorter term, another design goal is to allow for development in longer term. As a result, we separate our design components into two as shorter term(realized) and longer term future candidate(hypothetical) components, which we describe next in our design.

5.1 Design

Figure 4 reflects the trust model of our design, which is in line with our design principles; leveraging existing user base, containing short and long term components, and also makes use of existing testbeds. The design components are given in five groupings from A to E. A and B show the subject of our current implementation, that is, shorter term design components. D is an evolution of B, showing a future complex trust model, and C and E are candidate example design components that can enter the system in the future. The dotted arrows show the identity trust, or "attesting the who the user is", whereas the full arrows show trust for resources; slices and slivers, as in the case of Figure 3(a).

In B we adopt a trust model similar to SFA/GENI interoperability of Figure 3(a), that is, AMs trust authorities directly in creating slivers on it, which results in a model similar to a circle of trust. We depict one part of this circle of trust, which is the subject of our prototype implementation in Section 5.2, serving as a proof of concept of the design. We separate the design pieces B and D, and only implement B as a result of our design principles (P3) and (P4).

Following our principle (P1), we choose to have all authorities join InCommon federation, which is depicted at A. There are dotted arrows from every authority at B, to every InCommon participant at A, such as university IT

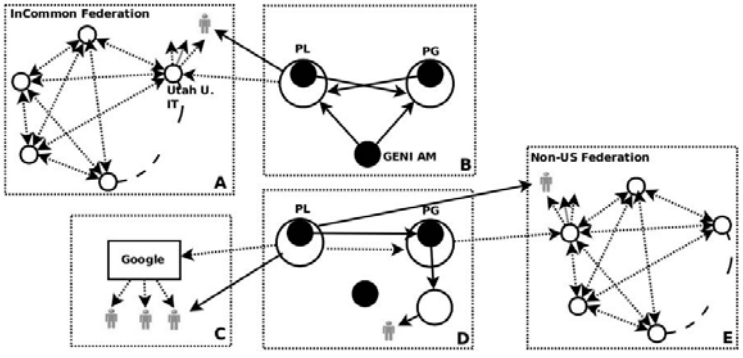


Fig. 4. Design trust model components; prototyped: A and B, future evolution of B: D, future additions: C, E

department(circles). We only show the dotted arrow from PL authority to one IT department, for illustration. The full arrow from PL authority to a user at Utah exemplifies an authority assigning slices to users from outside its domain. This happens by PL authority first authenticating the user by constructing the trust path through dotted arrows from itself to the user. After verifying the user is a student with appropriate affiliation and other properties, the authority can decide to assign him a slice, establishing a direct trust with the user. Using this slice, the user can use the resources at all aggregates, since it has a trust path from every aggregate to itself. Allowing the users to be authenticated by testbeds through SAML protocol also enables ease of entry and use for users, eliminating the hurdles of joining a domain again, and having to deal with public cryptography authentication.

While the user base of Incommon, that is the academic users, is of interest to compute testbeds today, in the future the users from other federations or domains can be interesting, as well. We depict this with the design part C where the authorities could authenticate the Google users, as well, possibly using a FIM solution such as OpenID [12], as discussed before. Such a scenario will be most interesting if the user base of Google is relevant to testbeds user base, and the set of published attributes by Google are relevant to testbeds.

The circle of trust model between authorities and AMs seen at B has a drawback in scaling with large number of federation participants, as mentioned before. Therefore, one future trust model can look more like a distributed model, discussed in Section 3. In such a model, the user authentications and slice authorizations can arise from indirect trust relations, examples of which are depicted in the design component D. First of all, all the arrows in B also exist in case of D, which we do not depict for simplicity. The indirect trust relations are generally from an authority to authority or from an AM to AM. As an example, PL authority trust all the IT departments that PG authority does, which is shown in the figure with dotted arrow between authorities. E shows a hypothetical federation similar to InCommon, but out of US. While PG joins this federation,

again having arrows from PG to every participant at E, PL may not join this federation since not all authorities can be expected to join all federations when the number of federations gets high. Instead, PL authority can get to authenticate a user at E, as shown in the figure, by following the dotted trust path that passes through PG authority, and then can create a slice for that user, as shown by the full arrow from PL to that user. Similarly, PL AM can trust all the authorities that PG AM trusts for creating slivers for, shown by the full arrow between AMs. Then, a user of a third authority, as seen in figure part D, can construct a trust path not only to PG AM, but also the PL AM following the full arrows. It is important to note that the trust model at D can get arbitrarily complex, which will be guided in the future by the reference trust models, examples of which are discussed in Section 3. Again, the software solutions for realizing such complex trust models are systems like ABAC which have a formal policy component, and having software implementations are in development to fit the future needs.

5.2 Implementation

Our prototype implements the trust model of Figure 4, design parts A and B. This implementation adheres to our principle of minimal software and standards requirements, and realizing shorter term design components. It consists of three nodes intended to mimic the operation of Utah University IT server, PlanetLab authority server and the GENI aggregate manager server in a federation. Figure 5 shows the software components at these three machines, and also the interaction between them that occurs during a Utah student creating a slice and allocating resources at AM using that slice. Utah student's actions for requesting a slice at PL authority are shown with steps 1 to 5, followed by user's operations on the AM, using that slice, with steps $1'$ to $5'$. There is an intermediate slice creation step, performed out of band, by PL operators, which is not depicted in the figure. All servers are set up from scratch in our a prototype, but IT server should be in place, so in a real deployment there is no need to create those components from scratch. The AM and authority servers are unique to our prototype in that there is Shibboleth to AM API inter-connection in the former, and Shibboleth installation for slice creation in the latter.

Since Utah University is a part of InCommon federation, its IT machine should run the SAML SSO IdP protocol, and we use the Shibboleth IdP software to implement it. We choose Shibboleth because it is the only solution supported by the InCommon support team currently, and it is the most mature solution for this protocol. We note that there are less stable alternatives of Shibboleth such as OpenSSO(Sun) and SimpleSAML, which are open source, and can support other protocols, such as OpenID, as well. SAML tokens represent a widely adopted format for authentications and authorizations among many FIM systems. The database and LDAP components are components which are generally used by such deployments to hold user names and passwords, and the user attributes, used by the IdP to keep user data. We use a database table called "users" to hold three user attributes; name, university ID and email.

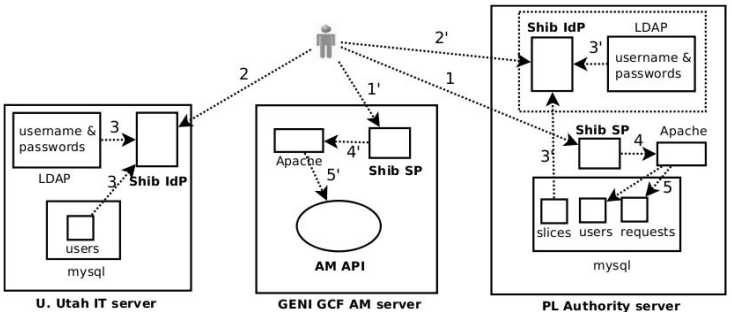


Fig. 5. Prototype implementation of design trust model parts A and B, using Shibboleth for authentication and authorizations

Our experience with Shibboleth software shows that not only the federated authentications but also slice authorizations can be realized with it. Therefore, we make use of Shibboleth IdP at the authority server, as well; to federate the user's slice related attributes, following our principle (P2), for minimal software and standards requirements in a federation. Shibboleth has security token format and the SSO protocol for exchanging tokens, defined by SAML standard. Our design inherits those as part of its minimal common schema, for compatibility between testbeds.

The slice request steps are as follows. At step 1 the user visits the PL authority web site slice request page. Shibboleth SP software intercepts the request before Apache web server, and returns the user a list of supported IdP's in a selection web page. The user selects his/her home organization, which is Utah University, and subsequently redirected to Utah University Shibboleth login page shown as step 2. Then at step 3, the user is authenticated with university user name and password, and his/her attributes, such as email, etc, are checked by the Shibboleth IdP at LDAP and database. If the user authentication is successful, user is redirected back to PL authority server, where it is allowed to access slice creation web page, as shown by step 4. The user requests the slice by filling out a form, specifying the slice name and its description. Also, if it is the first time the user is requesting a slice at PL authority, he/she creates a new user name and password with PL. User and slice request are stored in users and requests tables at the authority, shown by step 5. A new account at authority is created because it is required during slice operations in further steps $1'$ to $5'$. This is an engineering decision in order to keep the trust model simple as SAML SSO, with the assumption that a user gets slices from at most one or a few authorities in a federation, hence does not need to remember many passwords.

Once the slice request is stored at the authority, there exists another step which turns the request into a slice and records it in the slices table at the authority. In PlanetLab, this step is a human decision process, where the operators decide if the slice should be created, considering requester, slice purpose, etc. For now we also assume a human decision, but this step could be automated, as well.

This step stores a record in slices table for the user, which can include a set of privileges in the form of operation names, set by the authority, determining what operations the slice can perform on an AM; such as *Delete*, *Start*, *Stop*, etc.

The steps by which the user operates on the AM are similar, and shown by $1'$ to $5'$. This time the user's slice privileges are passed by the IdP at PL authority to the SP at GENI AM server, as user logs in PL authority successfully. Depending on slice privileges passed, the user can perform some operations at AM by setting parameters with a web form, which invokes the operation at the AM daemon, shown by step $5'$.

5.3 Evaluation

We verified the correct processing of our implementation by using the AM API of SFA to perform real AM operations at step $5'$, where this API is also backed by GENI interoperability effort. The original AM API has certificate based authorization for operation requests, therefore, we have changed code so that AM API accepts operations from local web server, as well, which uses attributes passed by Shibboleth for authorizations. Our total code in AM server is a few hundred lines of code, together with web user interface which is in PHP. Our code at authority server, on the other hand, implements first an administrative page to modify and view the slices table content, second, a public page which implements the logic to fill requests and users tables based on user attributes obtained from Shibboleth, which are totally about 500 lines of code. We verified that slice requests and AM operations can be performed just by a web browser with no special requirements, which is important for usability purposes. In addition, we also developed a programmatic interface which allows the user to perform the AM operations on the command line. This is useful for users to operate on many AMs together. This tool is in python, and is about 200 lines. It gets command line parameters; user name and password, authority and destination AM addresses from user, and then performs AM operations in an automated fashion, by running steps $1'$ to $5'$ using automated web requests that use https connections and web cookies.

6 Related Work

SHARP [5] introduces a computational economy based on bartering using signed tickets expressing compute resources. An implementation of SHARP is used by ORCA testbed. SHARP has complicated features such as multi-hop and amount specific resource delegations, which are not in minimal requirements our design contains and easy enough for every federation participant to adopt.

Contracts in declarative languages [16] explores how to specify agreements in formal way. This also belongs into the longer term set of technologies, complementary to the ABAC in providing resource allocation policies, and complementary to SHARP as a formal logic component in specification of resource exchanges between peering partners in a federation.

Similar to SFA, public cryptography for user authentications is used in grid systems, too, such as OpenScience Grid [11]. Our implementation is based on user name and passwords for better usability and ease of joining for users. MyProxy [2] increases usability by managing user keys and credentials with secure proxies, but still has a barrier to entry for new users. CILogon [4] allows logins to grid through InCommon, leveraging academic user base, similar to part of our design.

AM API and the cloud computing API such as Amazon EC2 [1] show similarities. In the case of RSPECs however, the cloud computing the resource specifications are generally simple instance types but testbeds have more complex RSPECs. There is still not agreed on a common RSPEC format among all testbeds and federation efforts.

7 Conclusion

In this paper we have analyzed the testbeds and federation efforts of today with the trust models formalism. We showed how they relate to each other and fit into a bigger picture of federation of testbeds. We identified the contemporary challenges, which range from technical challenges to human related complications. While most of the testbeds today use certificate based authentication and authorization, we point to the advantages of utilizing user name and password based authentication, which can increase ease of use and lower barrier to entry to the systems in medium term. We separate our design into two stages, shorter term and longer term, and conclude that the two can have different system requirements, and the system implementation should evolve as the real trust relations are being established, by time. Moreover, we leverage the existing trust relations established in other systems such as InCommon, which otherwise would be hard to establish from scratch. Trust model analysis allows us to see clearly the trust requirements of federation. We believe our formalism to analyze trust structures of different system will be valuable in deciding the establishment of further inter-organizational trust relations.

Our prototype implementation proves that the shorter term federation design can be realized with existing software solutions. We implemented our design with minimal software requirements, where we used Shibboleth IdP and SP both user authentications from other domains, and also resource authorizations. Our next steps will be first, to deploy our prototype as working instances at PlanetLab and ProtoGENI authority and aggregate managers. Since many universities and research labs already joins Incommon, and has experience with Shibboleth, and also our prototype effort has resulted in complete instructions for deployment, we believe it will be easier to adopt. Another next step is to utilize other FIM systems, such as simplesaml, which can increase the federation's outreach to more external users with using support for protocols like OpenID.

Acknowledgements. This work was sponsored by the GENI Project Office at Raytheon BBN Technologies in an effort to evaluate existing frameworks

for providing identity management for the GENI federation. GENI project was funded by National Science Foundation (NSF). We thank Aaron Helsinger for his invaluable comments and guidance throughout the project.

References

1. Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/
2. Basney, J., Humphrey, M., Welch, V.: The myproxy online credential repository. Softw Pract. Exper. 35(9), 801–816 (2005)
3. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: Proceedings of the IEEE Symposium on Research in Security and Privacy Technical Committee on Security and Privacy, Oakland, CA, IEEE Computer Society Press (1996)
4. CILogon: Secure access to NSF CyberInfrastructure, http://www.cilogon.org
5. Fu, Y., Chase, J., Chun, B., Schwab, S., Vahdat, A.: SHARP: an architecture for secure resource peering. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), Bolton Landing, NY, USA, pp. 133–148. ACM (2003)
6. GENI: Global Environment for Network Innovations, http://www.geni.net
7. InCommon Federated Identity and Access Management, http://www.incommonfederation.org/
8. Jim, T.: SD3: A trust management system with certified evaluation. In: IEEE Symposium on Security and Privacy, pp. 106–115 (2001)
9. Li, N., Mitchell, J.C., Winsborough, W.H.: Design of a role-based trust-management framework. In: IEEE Symposium on Security and Privacy (SSP 2002), Washington, Brussels, Tokyo, pp. 114–130. IEEE (2002)
10. Liberty Trust Models Guidelines, http://www.oasis-open.org/
11. Open Science Grid: A national, distributed computing grid for data-intensive research, http://www.opensciencegrid.org/
12. OpenID Decentralized authentication protocol, http://openid.net
13. Peterson, L., Bavier, A., Fiuczynski, M., Muir, S.: Experiences Building PlanetLab. In: Proc. 7th OSDI, Seattle, WA (November 2006)
14. Peterson, L., Ricci, R., Falk, A., Chase, J.: Slice-Based Federation Architecture. In: Ad Hoc Design Document (July 2008)
15. ProtoGENI: Prototype implementation and deployment of GENI, http://www.protogeni.net/
16. Reeves, D.M., Grosof, B.N., Wellman, M.P., Chan, H.Y.: Toward a declarative language for negotiating executable contracts (June 23, 1999)
17. Security Assertion Markup Language (SAML) v2, http://www.oasis-open.org/specs/#samlv2.0
18. Shibboleth federated identity management system, http://shibboleth.internet2.edu/
19. Wallom, D., Spence, D., Tang, K., Meredith, D., Jensen, J., Trefethen, A.: A trefethen: Shibgrid, a shibboleth based access method to the national grid service (2007) (submitted to ahm)