Milestone DigitalObjectRegistry: Security requirements for implementing Clearinghouse and Registry Services.

1. Introduction

The Registry design, submitted to meet our first milestone (May 12, 2009), will be implemented and deployed over the public Internet within the next few months. The Registry will be used to federate clearinghouses, and other relevant data, across all the clusters. We will start by providing access to the ProtoGENI clearinghouse information, adding additional GENI clusters and projects as we gain access to them, and get cooperation from the relevant groups. This document highlights the security requirements we believe are important for this federation process.

2. Security Study

The well-known security requirements for any network-based system, including ours, are as significant here as elsewhere. These requirements include issues such as ensuring message integrity and confidentiality, dealing with DoS attacks, and keeping accurate traffic logs for auditing purposes. To be successful as an experimental and prototyping infrastructure, the GENI environment must be trusted and reliable. Experiments must be repeatable, so the state conditions must be well known and must exclude any unwanted interference, both malevolent and unintentional. However, for federation in the GENI environment our primary concerns are the trust model, distributed authentication, user autonomy, and privilege revocations.

The standard PKI model, where the authenticating server matches a client's private key with the published public key of the user without actually having the client pass the private key to the server, is widely adopted for authentication and session encryption purposes. This authentication model is almost always augmented with a trust model. An endorser (aka issuer), who knows about the client (a user), certifies the user's public key and other user related information by signing, encrypting the data to be signed with endorsers private key. The user's public key, other user information, signature, and the issuer information are captured in a byte stream known as a digital certificate for the user. As part of the authentication process, servers process the client's digital certificate and may approve or reject the request based not only on the matching keys, but also on the trust the server has in the issuer of the client certificate. That is, *trust* is assumed to be transitive. This assumption, if applicable to a given system, allows servers to trust unknown clients if the servers trust the issuers of those client certificates.

There are several management issues in this model. For servers to trust clients based in part on certificates, servers will have to know, and manage their trust in, the certificate chains. Additionally, if a client or an issuer turns rogue, then all the systems need to be notified to remove the rogue certificate from their trust store. There is usually lag time in this process, and the delay could compromise the security of the system very quickly. All servers, even those belonging to a single large distributed system, need to manage their own trust store. Managing a set of certificates, keeping that set up to date across multiple servers is a very big challenge, especially in a project as large as GENI.

3. Proposed Security Requirements

We propose using the Handle System (Sun, Lannom, & Boesch, 2003) (Sun, Reilly, & Lannom, 2003) (Sun, Reilly, Lannom, & Petrone, 2003) to offset the challenges that exist in the current PKI and trust model. Handles are of the form <prefix> / <suffix> (CNRI, 2008). Each organization in our recommended model will have its own prefix and have administrative permission for the handles under that prefix. In the recommended model, any given user certificate will include the handle assigned to the user. When, and if, such certificates are created, a handle is also generated and associated with relevant information, including the public key, of the user by the appropriate administrative agency. Any server processing a client's request would adhere to the following processing model:

- Client forwards its certificate to the server as part of a request
- Server retrieves the handle identifier from the certificate
- If the handle belongs to the set of prefixes assigned to GENI, then the server resolves (CNRI, 2006) the handle and gets the client's public key
- Server matches the client's private key with this public key

If matched, the client is authenticated and trusted. The fact that the handle belongs to a prefix allotted for GENI organizations implies that the handle belongs to a valid GENI space. The administrative rights of the prefix are fixed in such a way that only appropriate agencies can create handles under that prefix. In the case of user handles, the employer organization may be the only authority permitted to create the user handles. The same approach could be applied at the organizational handle level, e.g., cluster handle creation would be restricted to the GPO. This constraint on handle administration ensures that if the client's key matches the public key in the handle, and the handle belongs to the GENI prefix space, the user can be trusted.

This approach relieves the servers from maintaining their trust store and requires only that they know the range or "set" of prefixes that can be trusted. The handle-based approach, however, does not preclude the use of certification authorities and chains of trust in the normal sense. The client certificates may still be nested within an organizational certificate, which in turn is nested in the responsible cluster organization's certificate and so on up to, in this case, the GPO. The chain of certificates need not be maintained at each server. Any server may instead simply resolve the handle specified in the certificate and verify that the handle belongs to the set of prefixes, and that the keys in the handle match the keys in the certificate.

To simplify management of the prefix set for the servers, the prefixes could be packaged together. In the existing GENI setup, there is the GENI program office (GPO), there are clusters under the GPO, there are organizations within those clusters, and so on. For example, prefix 45678 could be assigned to the GPO; prefixes 45678.1, 45678.2, etc., could be assigned to clusters, prefixes 45678.1.1, 45678.1.2, etc., could be assigned to organizations participating in cluster 1, and so on. Prefixes assigned in this manner allow servers to trust handles belonging to a prefix that begins with 45678 without having to worry about managing a list of prefixes. A similar setup is used for the organizations using the ADL Registry, a US DoD registry that facilitates the reuse of learning and training materials for the military. It is important to note that prefixes created in this manner are no different than the prefixes that do not follow the dot-separated scheme. As always, the administrative privileges can still be held at the prefix level and/or the individual handle level, with no change to handle resolution. A GENI-wide registry/clearinghouse may hold a global trust policy with the corresponding prefix, 45678 in this example, or the set of prefixes, for servers to trust. This is one of the GPO's objectives - to put global policy at the clearinghouse level. However, there may be cases where individual servers have more restrictive policies than the GPO requires. For example, a server might decide to only trust users from two specific organizations. In those cases the server may be configured to maintain the list of handle prefixes for the trusted organization, perhaps in the form of a local policy held in a local clearinghouse.

Privilege revocations can easily be achieved by having the responsible organization delete the public key from the (rogue) user handle record, or by deleting the entire handle record, and any subsequent attempts by the rogue user to get authenticated by any GENI server would fail because the server would not be able to resolve the rogue user's handle to acquire the public key to match the corresponding private key the user claims to have during a PKI challenge-response sequence. All servers are synchronized with any revocation (deletion of handles) without actually having to manage any trust store, and the revocation is immediately effective.

In cases where an entire organization is categorized as rogue, the prefix for that organization (along with its handles) can easily be removed from the Handle System. Any handle resolution attempt, (comparable to a DNS resolution (CNRI, 2006)) by a handle client begins with a request to the Global Handle Registry® (GHR) (equivalent to DNS top level servers, com, net, etc.) to obtain the network location of the handle server responsible for that handle. Using the setup of prefixes from our earlier example, with the rogue organization running a handle server for the prefix 45678.1.1, response to requests for handles under the prefix the 45678.1.1 prefix can be denied at the GHR. This means that any subsequent resolution requests for those handles made by the servers, integrated with a handle client, will result in a HTTP 404 equivalent error. Failure to resolve a user handle by a server implies an authentication/trust failure. The revocation is effective immediately without the servers updating their trust store.

There are two latency issues. The first, which will not be addressed at this time, is the amount of time required to determine that a user or organization went rogue and take the appropriate action. The second is caching; that is, clients cache handle records and have a time-to-live (TTL) period. To deal with this issue, it is necessary to bring the TTL to appropriate level, or make handle clients always perform authoritative requests which ignore the caching and ensure that any handles deleted from the system are not treated as valid due to caching. In any case, the fact that the user handles must exist with the right public key ensures explicit trust capability quite easily when compared with the signature approach practiced commonly and described above.

Other security issues such as spoofing and man-in-the-middle attacks are also addressed by our implementation. Since it is required (and ensured during the authentication stage) for a user claiming a certain handle to have the private key that matches the public key for that handle that is stored in the handle server, authentication servers will know when a user creates a spoof certificate with a valid handle identifier embedded in the certificate; the public key generated by the spoof user could not have been associated with the claimed handle in the handle server managed by some (valid) GENI organization.

To deal with man-in-the-middle attacks, handle clients can request that the handle servers sign the resolution responses with the servers' private keys. A client can verify the server's signature by first getting the server's public key from the GHR and making sure the signature is made by the claimed server. A client can trust, after verifying, the response from the GHR because that response is signed by the GHR's private key, the public key of which may be accessed out-of-band. Out-of-band methods for getting the GHR's public key are currently being formalized.

Finally, handles also provide a way to store user credential information along with the trust and authentication information (CNRI, 2007). Storing the related information at one place brings cohesiveness to the system, thereby easing the management of those information items.

4. Clearinghouse Security Implementation

In our Federated Clearinghouse implementation, handle prefixes will be assigned to organizations as outlined above, and handles will be created for users from those organizations. The handle records will contain the users' public keys and credential information. The user certificates will store the handle identifier. To incorporate existing users at ProtoGENI, the gid stored in ProtoGENI user certificates will be used to compute the appropriate handle to assign to the users. Thus, when the Registry encounters certificates with no handles it can use a deterministic computation to identify the handle.

The Registry will implement an SSL interface, and any client requests will first be authenticated and trusted. If the client is not authenticated or trusted, the connection is broken. If the client is authenticated and trusted, then the Registry application layer will process the request and will use the credential information stored in the user handle to proceed further with the request. The process flow is authenticate->trust->authorize->process.

Initially, CNRI will provide the hosting and administration of those handles, but later on, depending on our involvement with GENI, those handles would be migrated to corresponding organizations to manage, along with their handle servers. When migration occurs, our system may be comparable to systems such as Shibboleth, where servers maintain user records and authentication parties may rely on those Shibboleth servers. The distinction between Shibboleth and the Handle System is that the Handle System allows administration at the single user record (handle record) level. For example, users themselves, in addition to the organizations, may manage their own profiles, credentials, etc. (Allowing individuals to manage their profiles is a feature that OpenID systems are providing.) The uniqueness of the Handle System, which is infrastructural at it's core, is that it can be customized to build a wide range of applications systems.

5. Conclusion

The Handle System adds many advantages to the widely used PKI model. Relieving the servers from maintaining their trust store and keeping the store in sync, by harnessing the scalability (CNRI, 2006) and distributed nature (CNRI, 2007) of the Handle System, would be a very practical solution to the security challenges the GENI system is currently facing.

6. References

CNRI. (2006). *Handle Resolution*. Retrieved from The Handle System: http://www.handle.net/overviews/overview.html

CNRI. (2008). *Handle Syntax*. Retrieved from The Handle System: http://www.handle.net/overviews/handle-syntax.html

CNRI. (2007). *Handle Value Types*. Retrieved from The Handle System: http://www.handle.net/overviews/types.html

CNRI. (2006). *The Handle System and the DNS*. Retrieved from The Handle System: http://www.handle.net/overviews/dns.html

CNRI. (2007). *The Handle System Architecture*. Retrieved from The Handle System: http://www.handle.net/overviews/architecture.html

CNRI. (2006). *The Handle System Scalability*. Retrieved from The Handle System: http://www.handle.net/scalability.html

Sun, S., Lannom, L., & Boesch, B. (2003). *RFC 3650.*

Sun, S., Reilly, S., & Lannom, L. (2003). *RFC 3651.*

Sun, S., Reilly, S., Lannom, L., & Petrone, J. (2003). *RFC 3652.*