1783 Milestone ExptsSec: S4.c

University of Alabama, Tuscaloosa, AL 35487
Xiaoyan Hong, Fei Hu, Yang Xiao, Bo Fu, Zhifeng Xiao, Songqing Yue
Sept 23, 2012

This document serves as the deliverable for 1783 milestone ExptsSec: S4.c. Work presented includes analysis on the key usage and management in the current GENI implementation, and further security analysis on authentication.

1. Analysis on the key usage and management

Our work with the analysis on the key management and usage explored several use cases of ProtoGENI. Our analysis identified several issues in two areas, namely, the management of multiple SSH keys and the SSH tools for remote login. They are described below.

1.1 Investigation on the Management of Multiple SSH Keys

In order to allow access to the allocated resource, the Clearing House must have a copy of SSH public key of the experimenter placed on the resource. The experimenter needs to provide the corresponding private key when remotely accessing the resource. If for some reasons, such as that the experimenter thinks his private key has been compromised, or that the experimenter generates a new certificate without using the private key when the old certificate becomes invalid, the experimenter needs to regenerate a new SSH key pair. Here comes the problem of managing multiple SSH keys.

In Geni, the current authentication system with regard to key management works in the following way, using FLACK as the example. Each time when the user generates a new SSL certificate, a new public key is created. The user can choose to reuse the existing private key or not. If not reusing the private key, a new private key will be generated along with the new certificate. Then the experimenter needs to download and save the private key in the home directory. The public key is automatically uploaded to allocated nodes when a new slice is created. The user can also choose to generate the SSH key pair by himself. In this case, the user needs to upload the generated public key through Emulab web portal. No matter which way of creating SSH key pairs is utilized, all the uploaded, both automatically and manually, public keys will be saved in the local file .ssh/authorized_keys on the reserved nodes, so that the user can login into each resource using Emulab generated private key or his own private key.

It may become troublesome for the experimenter to maintain more than one private key in the local folder. Especially in the case where the experimenter has allocated resources with different certificates and those different certificates do not share the same private key. We have set up an experiment to illustrate the scenario. In the experiment we use FLACK to reserve a slice including a node denoted as Node1 with the certificate generated by the Emulab server. An SSH key pair (Public Key1 and Private Key1) along are generated. After successfully reserving the node, Public Key1 is uploaded to Node1 automatically, so that we can ssh to the node with Private Key1 which is saved in a local folder. Then we generate another certificate (Certificate2) not reusing the private key, so a new key pair (Public Key2 and Private Key2) are generated. Now in our local folder, we have two private keys in store. We then create another slice including a node as Node2. Both Public Key1 and Public Key2 are automatically uploaded to Node2; therefore, we can ssh to Node2 with both Private key1 and Private Key2. However Public Key2 will not be uploaded to those resources reserved before the generation of Public Key2 and Private Key2, so we can only ssh to Node1 with Private Key1.

Suppose the experimenter mean to use the new Certificate2 and the new key pair (Public Key2 and Private Key2) for all the experiments after they are generated, then our previous experiment shows two problems: first, Node1 does not have the new keys, using it is not possible; and second, Node2 has multiple keys, using Node2 can get into confusion. On the other hand, if the experimenter intends to keep both Node1 and Node2 usable, he needs to remember the mapping between the keys and the resources.

Our experiment is only a simple demonstration of having multiple SSH key pairs. If the number of private keys of a user increases, the situation would become worse. The user needs to clearly remember which private key to use to log into a resource, which would cause great inconvenience for the user.

The most serious drawback in this mechanism lies in the fact that all previously generated private keys can be used to SSH to nodes reserved afterward. Even if the user has noticed that his private key has been compromised and has already regenerated a new certificate along with a new SSH key pair, the compromised private key still can be used by the attacker to access resources reserved later by the user. In the experiment, even though we have created a new certificate with a new passphrase and a new key pair, we still can use the old certificate, the old passphrase and old private key to remotely log in to the host as long as the old certificate is still valid.

To summarize, the current management of multiple private keys can result in the inconstancy of the keys distributed at different nodes, also, the potentially unnecessarily holding of outdated keys. These are the open questions need to be dealt with.

Similar comments were given by Leigh Stoller in:
"https://groups.google.com/forum/#!topic/protogeni-users/TkGley_dlXU".

1.2 Investigation on Remote Login Tools to GENI Nodes

In Geni, all the allocated nodes for users can only be accessed via SSH. Theoretically, it is independent of which tool users are using to ssh in to the allocated hosts. Any ssh client can be used to achieve this, as long as the following information is correct: 1) the name of the node to login, 2) the username that is one's username with Geni, 3) the private key that corresponds to the public key to be loaded on the machine.

Currently there are three ways of allocating GENI resources in ProtoGENI: Flack, ProtoGeni and Omni. After the successful reservation of resources through these interfaces, the private key needs to be handled carefully so as to facilitate the later remote logins to the allocated nodes. If the user is using ProtoGeni, SSH keys are usually generated with the *ssh-keygen* command, and the user public key is placed in the home directory "~/.ssh/id_rsa.pub" before experiments.

Thus, after the successful reservation of the resources, the public key needs to be manually uploaded to the slice. If FLACK is used, the experimenter can request the Emulab server to generate an SSH key pair along and a SSH certificate, then download and save the private key in the home directory. The public key is automatically uploaded when a new slice is created. If the user makes the reservation with Omni then he needs to access the host from the machine where Omni is installed to get a copy of the private key. The SSH public key is automatically uploaded in Omni.

For all Linux and Mac, the built-in SSH command can be used to log in to the allocated nodes using the command*: ssh –i <private key location> <username>@<hostname> -p <port>*. Before that the private key file should be configured to have the right permission: *chmod 400 <private key file>*. For windows user, currently there are several options, such as Cygwin, SSH Secure Shell Client and Putty. Cygwin helps to set up a Linux-type environment including a built-in ssh client. Using Putty, the private key has to be converted to format Putty recognizes. In SSH Secure Shell Client, the user has to carefully follow its instruction to upload the private key to the client. By now, there is no other simple ways to remotely login to the allocated hosts in windows, let alone for other operating systems.

In win7, after successfully creating a slice and reserving a node using FLACK, the “ssh button” does not work with any mainstream web browsers, such as Explore and Firefox. Before clicking on the “ssh button”, the correct SSL certificate in pkc12 format has been successfully loaded into the browsers. And FLACK shows the public keys have been uploaded to allocated nodes to allow login with corresponding private keys. However, neither the “Visit button” nor the “SSH button” works. Further supports need to be provided to make it work not only in Mac but also in other operating systems.

In summary, two issues are identified. One is that it could be uneasy for users who login to GENI resources across multiple GENI interfaces and use built-in SSH tools in various operating systems. In addition, for FLACK, the browsers (tested Explore and Firefox) were not the cause of the not-working “Visit button” and “SSH button”, but operating systems were.

2. GENI Authentication

In this part, we studied and investigated the authentication mechanisms for Global Environment Network Innovation (GENI).

We investigated the authentication mechanisms employed by GENI projects. GENI projects mostly adopt traditional password-based and public-key-based authentication. We provide a few suggestions as follows:

Protection of certificates and credentials: In PlanetLab, an API can be called when the user's certificate is authenticated by the resource provider. In ProtoGENI, a user authenticates itself by presenting a credential/ticket issued by the resource provider from an earlier time. Therefore, if the certificate or credential is compromised by an attacker, the system's trust model will be in jeopardy. To this end, it is highly necessary to add protection mechanisms to key pairs and credentials in order to keep them from being compromised. To avoid misusage, either the SSL directory should be protected, or any other secondary authentication method should be implemented along with the existing one. For credentials, a dedicated server for credential management may suffice. With this credential server, users do not store credentials locally. Instead, users will need to download credentials from the credential server via a secure channel and then present them to the resource provider.