Distribution of security-sentive data

A. Montiel González[1] *and M. Pausch*[1]
[1]GSI, Darmstadt, Germany

A new system for authentication and authorization has been implemented to substitute the old Network Information System (NIS). This is based in the integration of Heimdal and OpenLDAP, which together combine remarkable security aspects and a reliable replication engine. In this infrastructure security items need to be installed on every *kerberized* network service. These are the so called keytabs, which hold one or more keys.

This system is going to be deployed by using Chef as configuration management system[1].

The process of creating a keytab requires a privileged user to invoke the Kerberos administration application and to access it with an authorized principal, generating a key to be installed on the system. Therefore, there is intervention by a human to enter a password. The use of Chef implies automation in such steps. Hence, it is required to automatically and securely distribute these keytabs and, eventually, any other security-sensitive data.

Two solutions to this problem are described here.

Wallet

Wallet[2] is a system for managing secure data for systems. In this system, secure data could be any file or Kerberos keytabs. It stores and retrieves these data, and can automatically create keytabs on demand. It applies ACLs, and audits trail of operations.

It is a client/server tool, where communication is accomplished via remctl. This protocol uses Kerberos for authentication and encryption. It runs simple commands on a remote host and returns the output.

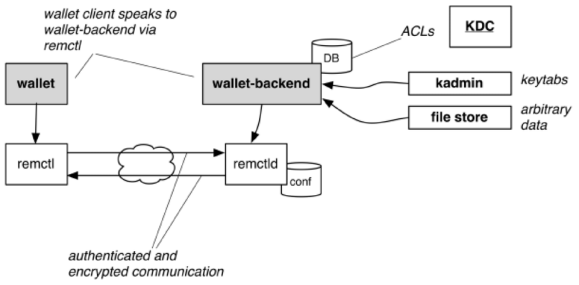

Figure 1: Wallet system. Figure taken from Ref[6]

As seen in Figure 1, the wallet-backend will keep the ACLs in a local database. Those ACLs control access to normal and administrative operations. The server also implements a front-end to the Kerberos kadmin, which retrieves keytabs. When retrieving a keytab, by default, the key is changed in the KDC and hence the invalidation of any existing keytabs for that principal. It also maintains an encrypted file bucket, where secure files are stored.

A wallet client will run on every node where secure data need to be provisioned. They must hold a keytab to communicate with the wallet server. For this purpose, a service principal for the wallet is created on Kerberos. This keytab is exported and copied onto all clients as part of the basic system. With this in place, and having configured the connection to the KDC and to the wallet server, the node could generate on demand any keytab and retrieve secure files, like the master key of KDC.

Chef

Chef[3] is a server/client based configuration management system for GNU/Linux. A *Chef-client* is running on every computer in the infrastructure that is managed by Chef. This Client uses RESTful communication to talk to the Chef-server, which is encrypted using public key cryptography. Since a cryptographic system is already in place when using Chef as configuration management system, this system can be used for encryption of other data.

In order to securely transport sensitive data from one node to another, the data are encrypted using a hybrid cryptographic system consisting of RSA and AES. The data are then uploaded to the Chef-server, from which the Chef-client can download and decrypt the data with the appropriate private key. The implementation of Chef allows easy extension in Ruby[4], which offers an interface to the OpenSSL[5] library that is used to handle the encryption of the data.

This approach should only be taken into account for small files, since big files can quickly lead to a lot of storage consumption on the Chef-server.

Conclusions

Both solutions are sound. The one extending Chef has been taken as first approach. The Wallet is a good alternative that prevents from relying on the Chef server reliability.

References

[1] C. Huhn et al, "Configuration Management Evolution in the GSI HPC farm", GSI Annual Report 2012.
[2] http://www.eyrie.org/ eagle/software/wallet/design.html
[3] http://www.opscode.com/chef/
[4] http://www.ruby-lang.org/en/
[5] http://www.openssl.org/
[6] http://jpmens.net/2012/06/25/streamlining-distribution-of-kerberos-keytabs-and-other-secure-data/