Table of Contents

Secrets for puppet

The problem

We need to manage secrets (passwords, certificates etc) on puppet managed machines. Using traditional puppet mechanisms isn't particularly helpful, since puppet compiles catalogs on the puppetmaster. In the case of something like hiera-gpg you cannot hide secrets from anyone else that has commit rights to the puppet manifests or the puppet masters. In our case we therefore cannot "trust" the puppetmaster with secrets. Yet we of course still need secrets on the machine, and it would be nice to have some sort of puppet management of files that needed secrets in them on our nodes.

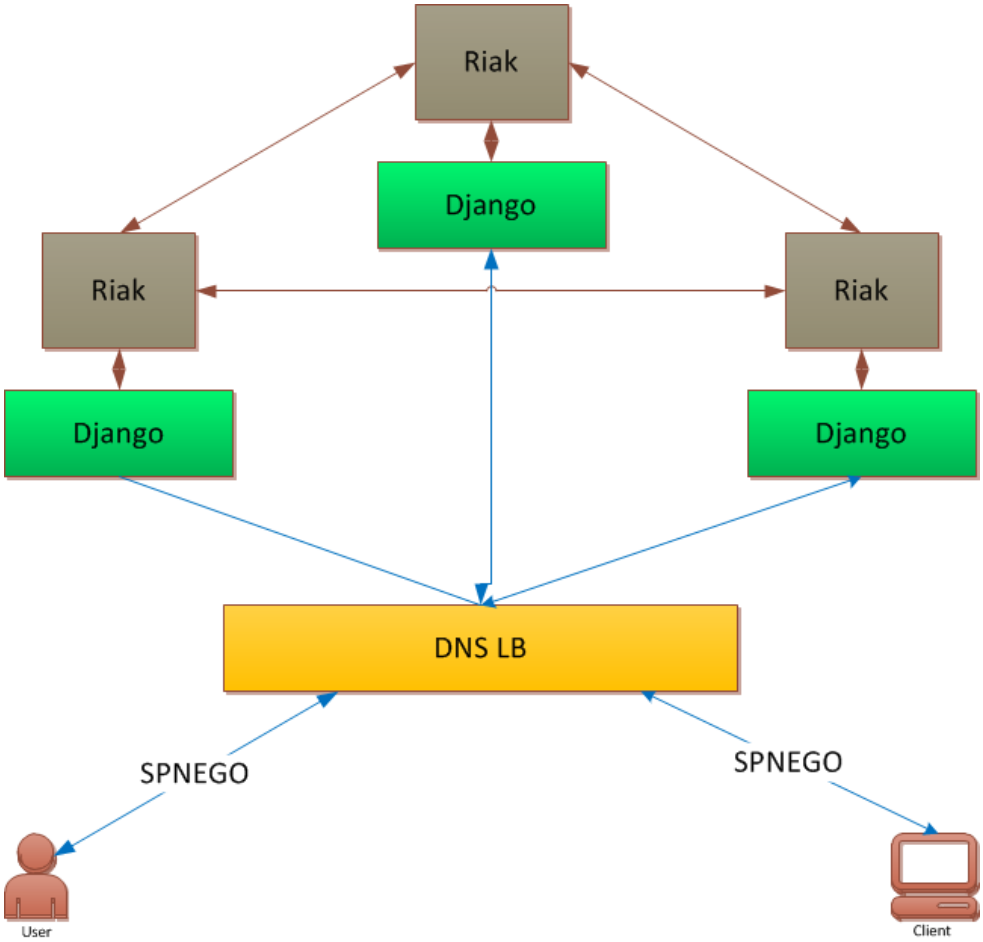
Solution summary

We trust the node credentials (kerberos host keytab), and we trust hostgroup membership since it is set at the ENC. We can therefore use an external store for secrets, in which data can be stored by hostgroup or hostname by arbitrary key. Users will be able to get/set/delete secrets k/v based on the service checking CERN data stores for host permissions. The host will then use its host keytab to pull secrets using spnego from the service and store them locally. Puppet integration will then allow them to use the secrets as file fragments for concat or even for file source in their puppet manifests.

Service description

teigi

Teigi is a django fronted riak datastore, with mod_auth_kerb for authentication, and links to things like landb, puppetdb and foreman to do authorization.



authorization sources

- puppetdb: checks the "k5logins" fact - if the machine is configured to allow the user to login as root, then the auth check passes.
- landb: checks the "responsible person" field. If the user matches, or if the field contains an egroup to which the user is a member, then the auth check passes.
- foreman: a faster cache of the landb "responsible" field

host authorization

The host authenticates to the service as host/fqdn using the host keytab. This means the host can be permissioned, for instances to allow the host to retrieve information stored for it.

workflow

storing secrets workflow

- client tool (REST client) sends blob with key to hostgroup or hostname endpoint. ie: "$ coffre store --key apache_cert --hostgroup punch/fore/balance/prod --file /tmp/secret_file"
- service authenticates users, then determines if they are allowed to administer this hostgroup (this is a question number 1, which I'll enumerate later)

- if auth passes, service uses it's gpg key to encrypt data, then stores it in riak.

downloading secrets workflow

- host client puppet type/provider (question 2) asks for secret by key
- service then has to lookup hostgroup (question 3)
- service finds the closest key (a la hiera) to serve the request, decrypts, then sends to the client
- client stores in (configurable) local directory, which is ensurable (cf q 2)

Gotchas / questions for workflow

1. There isn't currently a particularly good way to work out hostgroup ownership. Is it good enough to pick a host from the hostgroup and use those permissions? Or can we query something from foreman.

2. It has to be a type/provider as the result has to be a puppet resource. We need to be able to clean up old secrets when, for example, a machine moves hostgroup.

3. It has to be the service that does the hierarchy lookup, because we can't trust the client. With normal hiera, it would be the client that would make a request per element of a hostgroup. However we're not in the position to allow the host to assert what it's hostgroup is. The hostgroup list would have to be done via the ENC. In the background, the django app would be looking in riak keys for the hostgroup (ie compound dotted keys for hostgroup elems).

Integration with puppet

The most convenient way to manage files with dynamic content in puppet is with templates. This isn't going to work here though, since templates are compiled on puppet masters. Therefore to use secrets stored in this way within other files, people would have to use concat, and have a concat fragment using a local file source.

This topic: Main > TeigiVault
Topic revision: r1 - 2013-11-27 - BeJones