Plugin System Notes

What does the Plugin Manager Support Now?

- Supported
  - Daemon - It is made for the daemon.
- Unsupported
  - Controllers and Views - The way the front-end was designed using HAML hindered the dynamic rendering capabilities of Ruby on Rails. It is very obvious plugins for the front-end is not supported at this stage.
  - Models - No information available for now. At first glance, should not support.

Possible files affected?

- `config/earth-webapp.yml`
- `app/controllers/*.rb`
- `app/helpers/*.rb`
- `config/routes.rb`
- `app/views/*.haml`

Plugin system core and supporting files?

- Core files
  - `lib/earth_plugin_interfaces/plugin_manager.rb`
  - `lib/earth_plugins_interfaces/earth_plugin.rb`
- Supporting files
  1. `script/create_cert.rb`
  2. `script/sign_plugin.rb`
  3. `script/install_plugin.rb`
  4. `script/earthd`
- Current primary plugin directories
  - `config/certifcates/` - Where the OpenSSL certificates are.
  - `config/keys/` - Where the server's key pairs are.
  - `lib/earth_plugins/` - Where the plugins should be.

OpenSSL and how to test?

The biggest hurdle with the plugin system is the signature check that is done while loading and installing a plugin. There are 2 ways that can be done to get the signature of a plugin.

Before running, make sure you have `script/create_cert` and `script/sign_plugin` scripts updated by replacing the `require_gem 'termios'` to the following lines:

```
   gem 'termios'
require 'termios'
```

1. The RSP way:
   (a) Run `script/create_cert` to generate a certificate of your host system. (You will need to create directories `config/certificates` and `config/keys` in order this script to work.
   (b) Run `script/sign_plugin <plugin>` to create a `*.sha1` signature file.
   (c) Run `script/console` to check whether the signature is valid. (NOTE: At this stage, you should have created 3 files: `config/certificates/test_cert.pem`, `config/keys/test_key.pem` and `<plugin>.rb.sha1`)
      i. Run `signature = File.read(‘‘<plugin>.rb.sha1’’)`
      ii. Run `code = File.read(‘‘<plugin>.rb’’)`
      iii. Run `cert = OpenSSL::X509::Certificate.new(`
      `File::read(‘‘config/certificates/test_cert.pem’’))`
      iv. Run `cert.public_key.verify(OpenSSL::Digest::SHA1.new, signature, code)`.
      (You should get `true` as the result after running this line.)
2. The `L33T` way:
   (a) Run `openssl genrsa -des3 1024 > host.key` to generate the key file.
   (b) Run `openssl req -new -x509 -sha1 -days 365 -key host.key > host.cert` to generate a certificate from the key.
   (c) Run `openssl dgst -sha1 -sign host.key <plugin>.rb > <plugin>.rb.sha1` to sign the plugin with the key and digest it with SHA1 algorithm.
   (d) Run `script/console` as the same as the 2^{nd} half of the RSP way to verify the plugin was properly signed and digested.

Possible solutions?

- The general idea of a plugin is to automatically load the plugins from designated plugin directories and load them one by one as the system goes along. This can be achieved by recursively calling the `require` or `load` command. In order to do this, the plugin system should be smart enough to only work when there are plugins in the plugins directories.
- One of the problem Earth is that it does not support Controllers and Views plugins. This is obvious with the way the plugin manager was designed. So, a solution to this is to extend the plugin manager to accommodate such plugins as well. That means the the plugin manager needs to verify the signatures of multiple files, instead of only one file.
  – The sub-problem of this approach is that the Radial feature was implemented as part of ApplicationController and any plugin will fail when Radial is being used. So, the idea is to convert the Radial into a plugin as well.