FIWARE-Aiakos: Aiakos

Release

September 19, 2016

Contents

Aiakos is a server with public API to manage ssh/gpg public keys for the support user of each FIWARE Lab node.

This project is part of FIWARE.

CHAPTER 1

Documentation

GitHub's **README__** files provides a whole documentation:

1.1 FIWARE Aiakos

1.1.1 Introduction

Server with public API to manage ssh/gpg public keys for the support user of each FIWARE Lab node.

This project is part of FIWARE.

Any feedback on this documentation is highly welcome, including bugs, typos or things you think should be included but are not. You can use github issues to provide feedback.

Top

1.1.2 Overall description

Aiakos is a service developed to store the public keys corresponding to each FIWARE Lab node in order to secure the access to the virtual machines instantiated in the FIWARE Lab.

Each region of FIWARE Lab must generate and provided a SSH public key and a GPG public key. As a remainder, it is perfectly secure that people have access to public keys. Only the private keys must remain secret.

Top

1.1.3 Why a SSH key and a GPG key are needed

The new base images and all the images based on them, as the migrated GE images, have a support account. There are two public keys with an important role in the support account:

- a SSH public key: This is the only method to log in the virtual machine with the support account via SSH (the password access is disabled).
- a GPG public key: This key is used to encrypt the password of the support account. That password is generated inside the VM and after the encryption is printed to the console. The support account password allows to access the VM via console, which is useful if the network is down. Besides, the password is also needed to get full root privileges after connecting to the support account both using console and ssh client due to, for security reasons, sudo is configured for asking the password.

Another reason to require the support password and encrypt it is to make easy enforcing full control and audit of who have admin access to each VM. This is because the SSH public key is not specific of each VM, but the password yes. There are alternatives but more complex. For example, do not provide the SSH private key to the support staff but use an special designed ssh-agent that audit each access. This document describes the protocol used by OpenSSH's ssh-agent.

Top

1.1.4 Why each region must use their own public keys

The region staff team are responsible of the virtual machines instantiated on their servers. Therefore each region staff should have the control of who access the virtual machines for support purposes and set and enforce the corresponding policy. It is not possible if the public keys are shared among all the regions. Additionally, it is also extremely insecure and a problem when a region leaves the federation.

Top

1.1.5 How the keys are injected inside the virtual machines. Limitations

The support account is created by a script invoked at boot time. This script is named *activate_support_account.py* and it is available at GlanceSync support scripts. Inside this folder there is also documentation and all the material to create the base images with the support account.

The script tries to download the userdata file from the metadata server. This is something that cloudinit also do, but this script can do more attempts because it works in the background. However, if the metadata server is not accessible (e.g. because the network is down) the script will finally give up. In this case, it will use a failback, a public SSH and public GPG keys hardcoded in the images. The failback keys are used also if the user modify the userdata content and removes the part that provides the keys.

Of course, the private key pairs hardcoded in the images cannot be shared among regions for the same reasons it is recommended that each region has its own keys. However a procedure could be activated to request that the support password of a specific virtual machine is being encrypted using the public GPG key of the region asking for it.

Finally, it must be considered that the support account can be disabled by the user, who has full control of the virtual machine. For example, the support account can be deleted or its password changed. Also the script that create the support account can be modified or deleted. In those cases, the administrator looses the control over the support of the virtual machine.

Note that the script is run at boot time, but the support account and password only should be generated in the first boot of the virtual machine. However, the encrypted password must be printed at each boot, because the console logs are clean when the virtual machine is rebooted.

It is important to delete the support account before making a snapshot to be used as template of new images; indeed this is one of the steps than the procedure to create GE images do. Otherwise all the instantiated VMs will have the same password in the support account. The script try to detect this checking if the UUID of the virtual machine has changed, but anyway during a few minutes the password might remain unmodified.

Top

1.1.6 How to generate and use the keys

Software vs hardware solution

This document only describes how to generate and use the keys through a software solution, but it is also possible use a hardware device, as an smartcard. The use of hardware solution is not documented here because it may be specific of each product, although some information will be provided AS IS to people interested in an introduction and some references to start with.

The advantage of a hardware solution is that it provides better security. It also make possible a full control and audit, because with a hardware solution the person who uses the key cannot copy it. Most of the advantages of a hardware solution can be replicated using a dedicated host and an agent: this way support staff does not have access to the key and the cryptographic operations are delegated to the agent.

A hardware solution consists of a device that stores the private keys and does not allow extracting them. Most of these devices generate the keys by themselves and therefore nobody has a copy of the private key, while others allow storing an existing key. People do not need to access the private key to use it, because the operations involving the key (e.g. to sign a challenge) are delegated to the device.

Some hardware solutions

There is neither an only solution nor standard. Some devices are intended for SSH keys, other for GPG keys but also support SSH. Neither all devices are supported in Linux nor provided free/open source drivers.

- An OpenPGP 2.0 card can be used with GPG and also with OpenSSH through the gpg-agent.
- The gnuk project allows using some very cheap STM32F103 microcontrollers with a USB port (it installs a firmware supporting the OpenPGP 2.0 card specification). This option is less secure than and smartcard or a specifically designed USB-token but safer than a software solution.
- The OpenSC projects is about using smartcards and USB-tokens through PKCS#11/PKCS#15 with Linux. This project does not work with GPG due to GPG does not speak PKCS#11. However some devices might work with and old project (probably unmaintained) that do a bridge between PCKS#11 and GPG.

A very cheap solution (but not the more secure, most of the other devices are designed to resists more types of attacks, including analysing the power consume) is to use gnuk project with some STM32 devices. This software is designed for GPG keys, but the documentation explains how to use with ssh through an agent.

These links are provided as reference only. The solutions described, including the gnuk project commented before, have not been tested and therefore the information is provided AS IS, without any support.

Generating a SSH key

A public key can be generated from different ways, including the generation of SSH keys from the FIWARE Lab Cloud Portal. For more details about it, we suggest to follow the indications in the presentation Setting up your infrastructure using FIWARE Cloud between slides 19 and 23. A simple way to generate a ssh key is just running this OpenSSH command:

```
ssh-keygen -N "" -f support_key
```

The file support_key will contain the private key. The file support_key.pub is the file that contains the public key and have to be provided in to the Aiakos Service.

Generating a GPG key

A gpg key can be generated with the following command:

```
gpg --gen-key
```

It is not convenient to run this command in a virtual machine, because it needs a lot of entropy and the command will stop waiting for more information from /dev/random.

It is very important that the name of the key is *Fiware support <region>*. If the key name does not start with *Fiware support* it is not detected by the script that creates the support account.

The public key is exported with this command:

```
gpg --armor --output public.gpg --export "Fiware support"
```

The public.gpg is the file that must be provided in the Aiakos service. To decrypt a message just execute the following command:

```
gpg -d message_file
```

Where message_file is the file in which we have found the encrypted text (in our case, it should be the text in the log file in which we see the encrypted password).

Top

1.1.7 Obtaining the password of the support account

The support account password is generated inside the VM, then encrypted with the GPG public key and printed to the console. The console logs can be obtained by the owner of the VM or by an administrator using the command *nova console-log*

The following script (getpassword.sh) can be used to decrypt the password:

```
#!/bin/bash

export OS_AUTH_URL=http://130.206.112.3:5000/v2.0

cat <<EOF > extract.awk
/-----BEGIN PGP MESSAGE-----/ {cp=1}
/-----END PGP MESSAGE-----/ {cp=0; msg=msg $0}
cp==1 {msg=msg $0 "\n"} ; END {print msg}'
EOF
nova console-log $1 | awk -f ./extract.awk | gpg -d
rm extract.awk
```

Keep in mind that the environment variables:

```
OS_REGION_NAME
OS_TENANT_NAME
OS_USERNAME
OS_PASSWORD
OS_AUTH_URL
```

have to be correctly configured. You can now check that the environment is correctly loaded, perform:

To run the script just write:

```
$ getpassword.sh <UUID>
```

or

```
$ getpassword.sh <virtual machine name>
```

where the UUID is the UUID of the virtual machine.

Top

1.1.8 API Overview

To upload new/modified a gpg key to the server. You should send a POST like this:

```
curl --request POST \
    --url http://aiakoshost/v1/support \
    --header 'accept: text/plain' \
    --header 'content-type: text/plain' \
    --header 'x-auth-token: 201dd9a13de844db905cb4f617cbc17d' \
    --data '-----BEGIN PGP PUBLIC KEY BLOCK-----\nVersion: GnuPG v1\n\nmQENBFWnVCYBCADPeDMbTOkCM4MPbU
```

The result of this operation is a text/plain response with the generated key:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1

mQENBFWnVCYBCADPeDMbTOkCM4MPbUMvtbAtGbUDnH3AHyZCEZZuyjeExATfT0Au
-----END PGP PUBLIC KEY BLOCK-----
```

Please have a look at the *API Reference Documentation* section below for more description and operations.

API Reference Documentation

- FIWARE Aiakos v1 (Apiary)

Top

1.1.9 Installation

Using FIWARE package repository (recommended)

Refer to the documentation of your Linux distribution to set up the URL of the repository where FIWARE packages are available (and update cache, if needed). Currently, `http://repositories.lab.fiware.org/repo/rpm/x86_64`

Then, use the package tool to install `fiware-aiakos`:

```
$ sudo yum install fiware-aiakos
```

Configuration file

Although some options can be specified from the command line, as a general rule the use of a configuration file is preferable:

- `/etc/sysconfig/aiakos.yml` (when running system service)
- `{installation_path}/config/aiakos.yml` (when running manually)

Such configuration file is self-documented, so you will find a description of every configuration option there.

After installing and configuring the service, you can execute the service with the following command:

```
$ sudo service fiware-aiakos start
```

And to stop the service, run:

```
$ sudo service fiware-aiakos stop
```

In order to test the service is running, run:

```
$ curl http://localhost:3000/v1/support/example/sshkey
```

Top

SSH and PGP keys

The key files for aiakos are stored in the folder /opt/fiware-aiakos/lib/public/keys. The naming must be <region_name>.sshkey and <region_name>.gpgkey (lowercase is mandatory)

Top

Unit tests

The `test` target is used for running the unit tests in the component:

```
$ cd fiware-aiakos
$ grunt test
```

Top

Build

Use the script provided for generate the package for the OS used:

```
$ tools/build/package.sh
```

Top

Docker image

You can use this Dockerfile to launch/execute the Docker image and container:

```
$ docker build -t fiwareaiakos .
$ docker run -p 3000:3000 -d fiwareaiakos
```

If you want to get more information about the use of docker see How to use Aiakos with Docker.

Top

1.1.10 License

(c) 2015 Telefónica I+D, Apache License 2.0

Top