ganto.acme_tiny Documentation

Release master

Reto Gantenbein

October 04, 2016

Contents

CHAPTER 1

Introduction

This Ansible role will setup acme-tiny, a minimal ACME client for the Let's Encrypt certificate authority. It also automates the creation of certificate requests and renewal of certificates, including service restart after the certificates have been replaced.

1.1 Installation

1.1.1 Ansible Role

This role requires at least Ansible `v2.0.0`. To install it, run:

```
ansible-galaxy install ganto.acme_tiny
```

1.1.2 acme-tiny

As there are different ways to setup **`acme-tiny`** on the various distributions this task is not covered by the Ansible role. It has to be done manually prior to running the role.

For Gentoo users the role author provides an acme-tiny ebuild in the linuxmonk-overlay.

If there is no package provided by the distribution of your choice, it can also be installed from PyPi.

```
pip install acme-tiny
```

CHAPTER 2

Getting started

There are two "modes" how this role can be run:

- *Scheduler mode*: Role will request new certificate based on an existing certificate request, replace the old certificate and restart the affected service. Ansible should be run with dedicated minimally privileged user account (by default `acmetiny`).
- *Setup mode*: Role will run the initial setup for a new domain certificate such as create required directories, generate RSA key and certificate request. Further it will make sure that a dedicated user acount for the scheduler mode is created and install the necessary **`sudo`** rules for the service restart. Role has to be run with `root` privileges.

2.1 Prerequisites

2.1.1 Let's Encrypt Account Key

Before the role can be run to send certificate requests an account key has to be generated. This can be done with the official Certbot client. Make sure the key is converted into the correct format for **`acme-tiny`** as described in Use existing Let's Encrypt key.

Eventually store the account key in `/etc/ssl/acme-tiny/account.key`.

2.1.2 Web Server Configuration

When requesting the certificate **`acme-tiny`** will place a challenge file in `/var/www/acme-challenges` which has to be accessible through `http://<fqdn>/.well-known/acme-challenge` for every domain requested in the certificate. Make sure to point the DNS entry of the domain name configured onto the system running this role and to add a corresponding definition in your Web server configuration.

The following snippets are meant as an example. Depending on the Web server configuration they need to be slightly adjusted.

Apache 2

```
Alias /.well-known/acme-challenge/ /var/www/acme-challenges/
<LocationMatch "/.well-known/acme-challenge/*">
    Header set Content-Type "text/plain"
</LocationMatch>
```

Nginx

```
location /.well-known/acme-challenge {
    alias /var/www/acme-challenges;

    location ~ /.well-known/acme-challenge/(.*) {
        default_type text/plain;
    }
}
```

Lighttpd

```
alias.url += (
    "/.well-known/acme-challenge/" => "/var/www/acme-challenges/",
)
```

2.2 Example playbook

A minimal playbook which would run the `ganto.acme_tiny` role to request a SSL certificate would looke like this:

```
---

- name: "Request and setup Let's encrypt SSL certificate"
  hosts: acme_tiny
  gather_facts: False

  roles:
    - ganto.acme_tiny
```

2.3 Example inventory

When using the *example playbook* the host to run the role has to be added to the `[acme_tiny]` host group in the Ansible inventory:

```
[acme_tiny]
hostname
```

Obviously, the Default role variables might not be suitable for everybody. Especially the *acme_tiny__domain* variable needs to be defined individually. This can be done via Ansible host variables in `/etc/ansible/host_vars/<hostname>/acme_tiny.yml`.

If there are multiple certificates that should be managed with this Ansible role, the individual configurations could be defined in separate "domain" files (e.g. `/etc/ansible/vars/<domain>.yml`) and then passed with the Ansible `--extra-vars` argument to the playbook execution.

Such a variable file would look like this:

```
---
#
# acme_tiny role configuration for: mydomain.com
#

acme_tiny__domain: [ 'mydomain.com', 'www.mydomain.com' ]
acme_tiny__service: 'nginx'
```

CHAPTER 3

Default role variables

Sections

3.1 Basic configuration

acme_tiny__config_dir

acme-tiny configuration base directory.

```
acme_tiny__config_dir: '/etc/ssl/acme-tiny'
```

acme_tiny__challenge_dir

Directory accessible through a HTTP server used for temporary challenge storage.

```
acme_tiny__challenge_dir: '/var/www/acme-challenges'
```

acme_tiny__account_key

File name of the Let's encrypt account key relative to the *acme_tiny__config_dir*. For more information see *Let's Encrypt Account Key*.

```
acme_tiny__account_key: 'account.key'
```

3.2 Domain configuration

acme_tiny__domain

Domain for which certificate is requested. Value can be a single domain or a list of domain names (e.g. `[ 'example.com', 'www.example.com' ]`)

```
acme_tiny__domain: 'example.com'
```

acme_tiny__cert_name

File name of key, certificate request and certificate (without file extension). By default this will be set to the (first) domain name defined in *acme_tiny__domain*.

```
acme_tiny__cert_name: '{{ acme_tiny__domain[0]
                          if (acme_tiny__domain is iterable and not acme_tiny__domain is string)
                          else acme_tiny__domain }}'
```

acme_tiny__cert_dir

Directory name where key, certificate request and certificate are stored for this domain.

```
acme_tiny__cert_dir: '{{ acme_tiny__config_dir }}/{{ acme_tiny__cert_name }}'
```

acme_tiny__private_key

File name of the RSA key used for generating the certificate request. If key doesn't exist yet, a RSA key of *acme_tiny__key_length* bit will be generated under this name.

```
acme_tiny__private_key: '{{ acme_tiny__cert_dir }}/{{ acme_tiny__cert_name }}.key'
```

acme_tiny__key_length

Length in bit of the RSA key.

```
acme_tiny__key_length: 4096
```

acme_tiny__cert_request

File name of the certificate request sent to the Let's Encrypt certificate service. The certificate request will be generated using *acme_tiny__domain* if not existent.

```
acme_tiny__cert_request: '{{ acme_tiny__cert_dir }}/{{ acme_tiny__cert_name }}.csr'
```

acme_tiny__certificate

File name of certificate which will be retrieved from the Let's Encrypt certificate authority.

```
acme_tiny__certificate: '{{ acme_tiny__cert_dir }}/{{ acme_tiny__cert_name }}.crt'
```

3.3 Service configuration

acme_tiny__service

Name (or YAML list of names) of the service for which the certificate should be installed and which should be restarted after certificate replacement. Value must be one of the configuration keys defined in *acme_tiny__service_map*. A custom service can be chosen by redefining *acme_tiny__service_map* in the Ansible inventory.

```
acme_tiny__service: ''
```

acme_tiny__service_map

Configuration map for defining default role behaviour regarding individual services. For more information see *Custom services*.

```
acme_tiny__service_map:
  apache2:
    cert_format: 'chain'
    cert_directory: '/etc/apache2/ssl'
    restart_command: '/usr/bin/sudo -n /usr/bin/systemctl restart apache2'
  dovecot:
```

```
    cert_format: 'chain'
    cert_directory: '/etc/dovecot/ssl'
    restart_command: '/usr/bin/sudo -n /usr/bin/systemctl restart dovecot'
  httpd:
    cert_format: 'chain'
    cert_directory: '/etc/httpd/ssl'
    restart_command: '/usr/bin/sudo -n /usr/bin/systemctl restart httpd'
  lighttpd:
    cert_format: 'keycert'
    cert_directory: '/etc/lighttpd/ssl'
    restart_command: '/usr/bin/sudo -n /usr/bin/systemctl restart lighttpd'
  nginx:
    cert_format: 'chain'
    cert_directory: '/etc/nginx/ssl'
    restart_command: '/usr/bin/sudo -n /usr/bin/systemctl restart nginx'
  postfix:
    cert_format: 'chain'
    cert_directory: '/etc/postfix/ssl'
    restart_command: '/usr/bin/sudo -n /usr/bin/systemctl restart postfix'
```

acme_tiny__service_restart

Restart affected service after certificate has been replaced.

```
acme_tiny__service_restart: True
```

acme_tiny__cert_format

Certificate format. By default the format will be determined by the service name which will lookup the value from `item.cert_format` in *acme_tiny__service_map*. The value must be one of `plain`, `chain` or `keycert`. If *acme_tiny__service* is undefined or empty the format will be `plain`.

Parameter description:

plain Simply store the certificate base64-encoded under *acme_tiny__certificate*.

chain Same as `plain` but additionally generate a certificate chain including the certificates of the issuing certificate authority. The certificate chain will be stored in `<acme-tiny-cert-dir>/<cert-name>_chain.crt` depending on the values of *acme_tiny__cert_dir* and *acme_tiny__cert_name*.

keycert Same as `plain` but additionally generate a PEM file which includes the base64-encoded RSA key and certificate. The PEM file will be stored in `<acme-tiny-cert-dir>/<cert-name>.pem` depending on the values of *acme_tiny__cert_dir* and *acme_tiny__cert_name*.

```
acme_tiny__cert_format: '{{ acme_tiny__service_map[acme_tiny__service].cert_format
                            if (acme_tiny__service|d() and
                                acme_tiny__service in acme_tiny__service_map|d({}) and
                                "cert_format" in acme_tiny__service_map[acme_tiny__service])
                            else "plain" }}'
```

acme_tiny__cert_symlink

Create symlinks from the service configuration directory (`item.cert_directory`) defined in *acme_tiny__service_map* to the actual RSA key and certificate in *acme_tiny__cert_dir*.

```
acme_tiny__cert_symlink: True
```

3.4 User configuration

User account meant for running certificate renewal via this Ansible role.

acme_tiny__user_name

User name.

```
acme_tiny__user_name: 'acmetiny'
```

acme_tiny__user_group

Primary group of the functional user.

```
acme_tiny__user_group: '{{ acme_tiny__user_name }}'
```

acme_tiny__user_home

Home directory.

```
acme_tiny__user_home: '/var/lib/acme-tiny'
```

acme_tiny__log_dir

Log directory owned by *acme_tiny__user_name*.

```
acme_tiny__log_dir: '/var/log/acme-tiny'
```

acme_tiny__log_file

Log file defined in `~/.ansible.cfg` of *acme_tiny__user_name*.

```
acme_tiny__log_file: '{{ acme_tiny__log_dir }}/{{ acme_tiny__user_name }}.log'
```

CHAPTER 4

System configuration

4.1 File system layout

The role will setup a configurable directory layout to store the certificates and make them accessible by the services.

Example layout of the default configuration:

Server path	Ansible role variable
`/etc/ssl/acme-tiny`	*acme_tiny__config_dir*
`/etc/ssl/acme-tiny/example.com`	*acme_tiny__cert_dir*
`/etc/ssl/acme-tiny/example.com/example.com.key`	*acme_tiny__private_key*
`/etc/ssl/acme-tiny/example.com/example.com.crq`	*acme_tiny__cert_request*
`/etc/ssl/acme-tiny/example.com/example.com.crt`	*acme_tiny__certificate*

When *acme_tiny__service* is not empty and an additional layer of indirection through symlinks which will make the certificates accessible in a transparent way. For each service the role will create a `ssl/` subdirectory from where the actual certificates and keys are symlinked. Like this the CA could be changed easily without reconfiguration of the secured services. This behavour can be disabled by setting *acme_tiny__cert_symlink* to `False`.

E.g. For for Apache **`httpd`** this would look like this:

```
/etc/apache2/ssl/example.com.crt -> /etc/ssl/acme-tiny/example.com/example.com_chain.crt
/etc/apache2/ssl/example.com.key -> /etc/ssl/acme-tiny/example.com/example.com.key
```

For **`lighttpd`**:

```
/etc/lighttpd/ssl/example.com.pem -> /etc/ssl/acme-tiny/example.com/example.com_keycert.pem
/etc/lighttpd/ssl/ca.crt          -> /etc/ssl/acme-tiny/intermediate.crt
```

4.2 Service configuration

To secure a service the key and certificate have to be referenced in the individual service configurations. When using the symlinks created by the role this only has to be done once. Any certificate changes and even the change of a certificate authority can be easily handled by pointing the symlinks to a new target.

Note: The configuration of the certificates in the service configuration files has to be done manually.

4.2.1 Apache httpd

```
SSLCertificateFile    /etc/apache2/ssl/example.com.crt
SSLCertificateKeyFile /etc/apache2/ssl/example.com.key
```

- Upstream documentation: Apache Module mod_ssl

4.2.2 Dovecot

```
ssl_cert = </etc/dovecot/ssl/example.com.crt
ssl_key  = </etc/dovecot/ssl/example.com.key
```

- Upstream documentation: Dovecot Wiki: SSL

4.2.3 Lighttpd

```
ssl.pemfile /etc/lighttpd/ssl/example.com.pem
ssl.cafile  /etc/lighttpd/ssl/ca.crt
```

- Upstream documentation: Lighttpd Wiki: Secure HTTP

4.2.4 Nginx

```
ssl_certificate     /etc/nginx/ssl/example.com.crt
ssl_certificate_key /etc/nginx/ssl/example.com.key
```

- Upstream documentation: Module ngx_http_ssl_module

4.2.5 Postfix

```
smtpd_tls_cert_file = /etc/nginx/ssl/example.com.crt
smtpd_tls_key_file  = /etc/nginx/ssl/example.com.key
```

- Upstream documentation: Postfix TLS Support

4.2.6 Custom services

The *acme_tiny__service_map* configuration dictionary can be overwritten from the Ansible inventory to extend the definition with a new service or adjust the current behaviour. Each element has the service name as key and needs to define the following properties:

cert_format Certificate format. See *acme_tiny__cert_format* for valid options.

cert_directory Custom directory from where the certificate and key will be symlinked. See *File system layout* for more details.

restart_command Command which should be executed to restart this service (instance) as an unprivileged user. If this command contains **sudo**, a corresponding rule will be created for the *acme_tiny__user_name* account.

Example

Custom Ansible inventory definition for Pound:

```
acme_tiny__service_map:
  pound:
    cert_format: 'keycert'
    cert_directory: '/etc/pound/ssl'
    restart_command: '/usr/bin/sudo -n /usr/bin/systemctl restart pound'
```

4.3 Certificate renewal

After adding a new domain the role has to be run once with `root` privileges. Among other things this will create a separate user account `acmetiny` which can be used to schedule unattended certificate renewals.

Note: See *Example inventory* for an example how to create a role configuration.

Here an example of a **`cron`** job (`/etc/cron.d/acme-tiny`) which whould renew the certificate every month:

```
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

@monthly acmetiny /usr/bin/ansible-playbook -e @/etc/ansible/vars/mydomain.com.yml /etc/ansible/playb
```

CHAPTER 5

Copyright

```
ganto.acme_tiny - Setup and renew Let's encrypt certificates with
                  acme-tiny

Copyright (C) 2016 Reto Gantenbein <reto.gantenbein@linuxmonk.ch>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see http://www.gnu.org/licenses/
```

CHAPTER 6

Changelog

ganto.acme_tiny

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ganto.

6.1 ganto.acme_tiny master - unreleased

6.2 ganto.acme_tiny v0.1.0 - 2016-10-04

6.2.1 Added

- Initial release [ganto]

A

E