Self-signed certificates via Ansible for local testing with Nginx

August 24, 2017

Most of my servers are using TLS certificates to encrypt all traffic over HTTPS. Since Let's Encrypt (and certbot) have taken the world of hosting HTTPS sites by storm (free is awesome!), I've been trying to make sure all my servers use the best settings possible to ensure private connections stay private. This often means setting up things like HSTS, which can make local / non-production test environments harder to manage.

Consider the following:

1. I go to https://servercheck.in/
2. Browser sees I have HSTS enabled, so caches that fact and won't allow me to connect to http:// version anymore.
3. I build local environment at http://local.servercheck.in/
4. When I visit that URL, browser detects that 'servercheck.in' has HSTS, tries to upgrade my request to https://, then fails since I don't have a ~~SSL~~ TLS cert configured locally.

So, to fix that problem, I now generate local 'self-signed' certificates for my servers using Ansible. This way not only can I access websites with the same root domain locally that are in production, but I also make my local testing configuration more closely match production (there are many little things that can differ between http:// and https:// requests!). Self-signed certs still pop up browser warnings, but at least the site is accessible after you make an exception for a self-signed cert.

Generating the certificate

First, I need to tell my playbook about any self-signed certificates so I can reuse the variables in both the task that generates the certs, and in configuration that uses the certificate paths. So, I normally use a variable named `self_signed_certs`:

```
self_signed_certs:
  - key: /etc/ssl/private/server.key
    cert: /etc/ssl/certs/server.crt
```

I defined a path to where I want the certificate *key* stored, as well as the certificate itself. On Ubuntu, the paths above should work for most use cases. For Red Hat-derivative OSes, you can use `/etc/pki/tls/private/server.key` and `/etc/pki/tls/certs/server.crt`. For production, where I either have a 'real' cert, or have Let's Encrypt generate a free cert for my site, I set `self_signed_certs: []`.

Next, I need to create a task in my playbook that generates a valid certificate and key using `openssl`:

```
    - name: Create self-signed certificate, if configured.
      command: >
        openssl req -x509 -nodes -subj '/CN={{ vagrant_hostname }}' -days 365
        -newkey rsa:4096 -sha256 -keyout {{ item.key }} -out {{ item.cert }}
        creates={{ item.cert }}
      with_items: "{{ self_signed_certs }}"
```

After the playbook runs, there should be a keyfile and certificate, valid for the next year, in the `key` and `cert` paths.

Using the certificate in an Nginx server configuration.

In this example, I'm using the `geerlingguy.nginx` role to install and configure Nginx, and define virtual hosts (`server` directives). Using Apache should be similar, especially if you're using my `geerlingguy.apache` role.

```
nginx_vhosts:
  - listen: "443 ssl http2"
    server_name: "{{ ansible_hostname }}"
    root: "/var/www/example.com"
```

```
    ssl_certificate: "{{ self_signed_certs.0.cert }}"
    ssl_certificate_key: "{{ self_signed_certs.0.key }}"
    ...
```

You can also use the 'snakeoil' testing certificates in Ubuntu (see `/etc/ssl/certs/ssl-cert-snakeoil.pem` and `/etc/ssl/private/ssl-cert-snakeoil.key`), if you want to try avoiding the self-signed certificate generation, but I like having a cert that is more valid for my particular hostname so I can manage certificate exceptions in my browser more easily.

Further reading

- Generating self-signed OpenSSL certs with Ansible 2.4's crypto modules
- Fixing Safari's 'can't establish a secure connection' when updating a self-signed certificate
- Remove a single Certbot (LetsEncrypt) certificate from a server

nginx ansible ssl openssl tls certificate https security self-signed automation

Add new comment