Internal Certificate Authorities

One of the problems in the deployment of a Public Key Infrastructure is the need for the certificates to be signed by a third party, called a Certificate Authority. By using a trusted CA, different entities that communicate with each other can ensure that certificates in use are valid and genuine. A common solution to that problem is the creation of a "self-signed certificate" which can be used by various applications as any other certificate. Unfortunately, this is not a sufficient way to ensure validity of a given certificate in a distributed environment, where various services need to ensure trust without a human intervention because other nodes in the cluster don't trust the self-signed certificate.

There are various solutions that let you set up a Certificate Authority which then can use an automated API to receive Certificate Signing Requests and issue certificates, however there always is an issue of bootstrapping such a solution. To ensure that the communication between requesting host and the CA is confidential, you need to provide the service over HTTPS, which requires a set of certificates, which require a CA, and so on, and so forth. An alternative is to request a certificate in an already existing Certificate Authority and configure them manually on your own CA server, however this requires human interaction. A proposed solution to this problem is ACME, used for example by the *Let's Encrypt*, however this solution cannot be used with internal hosts, which still need to be protected.

The `debops.pki` role solves this problem by creating it's own set of internal Certificate Authorities, located on the Ansible Controller in the `secret/` directory (see debops.secret for more details). These Certificate Authorities can be used to bootstrap a new PKI environment, which can then be passed over to a stand-alone CA server located on the network. Alternatively, certificates signed by the internal CA can be used for internal communication between hosts in the cluster, and external certificates can be enabled for public services like websites or mail servers.

Layout of Certificate Authorities

By default, the `debops.pki` role creates two Certificate Authorities:

- a Root Certificate Authority which is used as the "trust anchor" by intermediate Certificate Authorities;
- a Domain Certificate Authority which issues certificates based on incoming CSRs from remote hosts;

The directory structure of the Certificate Authorities stored in the `secret/` directory on the Ansible Controller:

```
secret/pki/
├── authorities/
│   ├── domain/
│   │   ├── certs/
│   │   │   └── BDD39DBA82436F2685A67FABD01519B8.pem
│   │   ├── config/
│   │   │   ├── authority.conf
│   │   │   ├── openssl-request.conf
│   │   │   └── openssl-sign.conf
│   │   ├── database/
│   │   │   ├── index
│   │   │   ├── index.attr
│   │   │   ├── index.attr.old
│   │   │   ├── index.old
│   │   │   ├── serial
│   │   │   └── serial.old
│   │   ├── issuer -> ../root/
│   │   ├── private/
│   │   │   └── key.pem
│   │   ├── requests/
│   │   ├── signed/
│   │   └── subject/
│   │       ├── cert.pem
│   │       └── request.pem
│   └── root/
│       ├── certs/
│       │   ├── F6F915290E08AB1A804E4092A9FEA4C9.pem
│       │   └── F6F915290E08AB1A804E4092A9FEA4CA.pem
│       ├── config/
│       │   ├── authority.conf
│       │   ├── openssl-request.conf
│       │   ├── openssl-selfsign.conf
│       │   └── openssl-sign.conf
│       ├── database/
│       │   ├── index
│       │   ├── index.attr
│       │   ├── index.attr.old
│       │   ├── index.old
│       │   ├── serial
│       │   └── serial.old
│       ├── private/
│       │   └── key.pem
│       ├── requests/
│       ├── signed/
│       └── subject/
│           ├── cert.pem
│           └── request.pem
├── ca-certificates/
│   └── by-group/
│       └── all
│           └── root-ca.example.com.crt -> ../../../authorities/root/subject/cert.pem
├── realms/
│   └── by-host/
│       └── hostname.example.com/
│           └── domain/
│               ├── external/
│               ├── internal/
│               │   ├── cert.pem
│               │   ├── intermediate.pem -> ../../../../../authorities/domain/subject/cert.pem
│               │   └── root.pem -> ../../../../../authorities/domain/issuer/subject/cert.pem
│               └── private/
└── requests/
    └── domain/
        └── hostname.example.com/
            └── domain/
                └── request.pem
```

The incoming certificate requests are placed in subdirectories of the `secret/pki/requests/` directory. Each subdirectory is related to a Certificate Authority, on above directory tree you can see that a request has been uploaded from `hostname.example.com` host for the `domain` Certificate Authority.

The certificates are placed in subdirectories of the `secret/pki/realms/` directory. The intermediate CA certificate and root CA certificate files are symlinked in the same subdirectory as the leaf certificate, so that Ansible can copy their contents as regular files to remote host and correct certificate chains can be created in the PKI realm.

Security of an internal CA

The Certificate Authority is a very vulnerable element of the Private Key Infrastructure. Hosts that have a Root CA certificate in their system certificate store will trust any certificates signed by that CA and it's intermediate Certificate Authorities, therefore protection of the Root CA private key should be taken care of as soon as possible.

The default Root Certificate Authority private key can be found in:

```
secret/pki/authorities/root/private/key.pem
```

This key allows you to create new Intermediate Certificate Authorities as well as revoke existing ones (although enforcement of the revocation in the form or distribution of Certificate Revocation Lists or an OCSP service is not currently implemented). You should protect access to it by moving the file to a secure location (preferably an encrypted, offline filesystem) and replacing it with an empty `key.pem` file (otherwise the `debops.pki` role will replace the private key and regenerate all of the CA certificates).

Unfortunately, private keys of the Domain Certificate Authority and any other Intermediate Certificate Authority or "Service CA", which is a Root CA used to sign service certificates cannot be protected by taking them offline - the private keys are required to sign certificates. Therefore, it is strongly recommended to store the `secret/` directory encrypted, and use it on an encrypted filesystem during use. In DebOps, you can use the EncFS filesystem together with the **debops-padlock** script to keep the `secret/` directory encrypted at rest. You should make sure that access to the plaintext files in `secret/` is only possible when it is really needed by unmounting the encrypted filesystem as soon as possible after usage and that only programs which need access can read the files by setting up restrictions like Mandatory Access Control and compartmentalization/sandboxing, to avoid leakage of private keys.

The Certificate Signing Requests created by `debops.pki` Ansible role contain a random challenge password (different on each run) which is then checked on the Ansible Controller, and only the CSR with correct passwords are signed by the Certificate Authorities. This should prevent signing of Certificate Signing Requests modified by a third party, unless the challenge password can be intercepted (it's currently passed using environment variables).

If for any reason CSR signing cannot be completed, you will need to remove the `internal/gnutls.conf` and `internal/request.pem` files to re-initialize the certificate signing.