Keeto Documentation

Release 0.2.0-beta

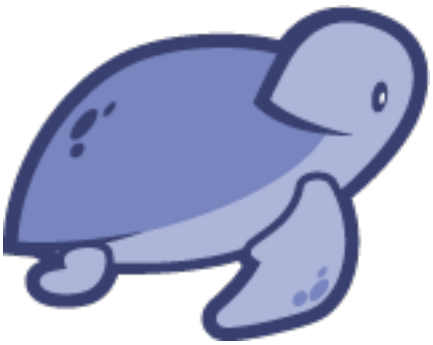
Sebastian Roland

Mar 14, 2017

Contents

1 Table of Contents **3**

1.1 About . . . 3
1.2 Concept . . . 3
1.3 Getting Started . . . 4
1.3.1 Prerequisites . . . 4
1.3.2 Installation . . . 4
1.3.3 Configuration . . . 5
1.4 LDAP Data Model . . . 6
1.4.1 SSH Server . . . 6
1.4.2 Access Profiles . . . 6
1.4.3 Key Provider . . . 7
1.4.4 Target Keystore . . . 7
1.4.5 Groups . . . 8
1.4.6 Keystore Options . . . 8
1.5 FAQ . . . 8
1.5.1 How can I exclude accounts from being processed by Keeto? . . . 8



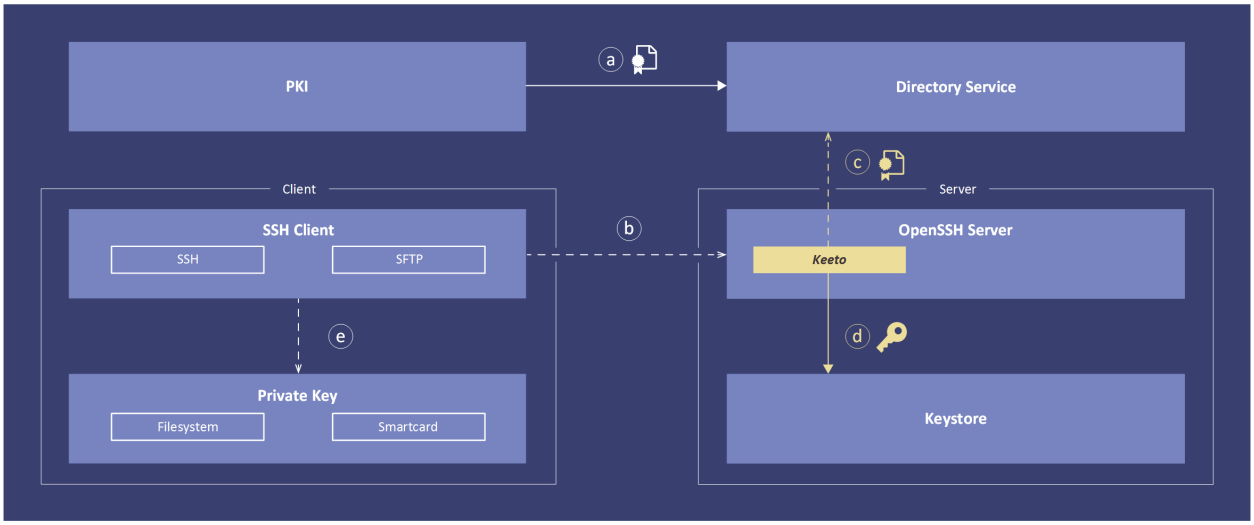
CHAPTER 1

Table of Contents

About

Keeto is a module for OpenSSH that enables profile-based administration of access permissions in a central LDAP aware Directory Service, adds support for X.509 certificates and handles the distribution of OpenSSH key material in an automated and secure manner.

Concept



The diagram shows a Keeto end to end flow. Each interaction is explained in the following. Note that flow **(a)** is periodically and all others are run on each SSH connection attempt.

(a) A Public Key Infrastructure (PKI) is responsible for managing the whole lifecycle of X.509 certificates. This managed X.509 certificates are distributed to a Directory Service where they can be used by Keeto.

(b) The SSH connection is triggered by a SSH protocol aware client such as PuTTY, FileZilla or WinSCP. An OpenSSH server receives the connection attempt.

(c) The control flow is passed over to Keeto which establishes a connection to the Directory Service and determines the current access permissions and retrieves relevant certificates for that connection.

(d) Keeto validates the X.509 certificates, extracts and transforms public keys and writes them to the appropriate authorized_keys file.

OpenSSH public key authentication is now taking over to authenticate the client against the freshly synced keys.

(e) The SSH client authenticates with the private key corresponding to the users X.509 certificate. Depending on the security requirements the private key can be held in software or hardware.

Getting Started

Prerequisites

The following packages are needed in order to run Keeto:

- OpenSSH >= 6.2
- Directory Service
- Syslog
- PAM
- pkg-config >= 0.9.9
- libConfuse >= 2.7
- libcheck >= 0.9.9
- OpenSSL 1.0.x
- libldap

Make sure those components are installed and configured prior setting up Keeto.

Installation

Source

Grab the source tarball from https://keeto.io and unpack/build. Note that the library installation directory for PAM modules (–libdir) differs for various architectures/distros. Consult the documentation of your distro to figure out the right path:

```
<user>$ tar xvfz keeto-0.2.0-beta.tar.gz
<user>$ cd keeto-0.2.0-beta
<user>$ ./configure --libdir=/lib64/security
<user>$ make
<user>$ make check
<root>$ make install
```

RPM

Grab the RPM package from https://keeto.io and install:

```
<root>$ rpm -i keeto-0.2.0-0.1.beta.el7.centos.x86_64.rpm
```

This installs the PAM modules and creates an initial configuration file keeto.conf as well as the authorized_keys and cert_store directories with the proper access permissions in /etc/ssh. Furthermore documentation is deployed.

Configuration

The following describes the configuration of the various components based on a installation from source. If an RPM package has been utilized certain steps do not need to be performed (see *RPM*). Also notice that the location of the samples directory differs for an RPM based installation. It can be determined as follows:

```
<user>$ rpm -qd keeto
```

Keeto

Copy the configuration file keeto.conf from the samples directory into the OpenSSH configuration root folder and adjust it to your needs. An explanation of the various options can be found within the file. As the config contains sensitive data make sure it is only readable/modifiable by a privileged user:

```
<root>$ SSH_DIR=/etc/ssh
<root>$ cp samples/keeto.conf $SSH_DIR
<root>$ chmod 600 $SSH_DIR/keeto.conf
```

Furthermore create a directory where the authorized_keys files of the users are placed and a directory for trusted CA certificates and CRL's:

```
<root>$ mkdir $SSH_DIR/authorized_keys
<root>$ chmod 711 $SSH_DIR/authorized_keys
<root>$ mkdir $SSH_DIR/cert_store
<root>$ chmod 700 $SSH_DIR/cert_store
```

Copy all trusted CA certificates and CRL's (optional) for verifying the user certificates into the cert store. Make sure the whole chain except the end entity certificates are present. If STARTTLS is used for the the LDAP connection also include the necessary certificates here. Finally create symlinks with:

```
<root>$ c_rehash $SSH_DIR/cert_store
```

OpenSSH

Make sure at least the following options are reflected in your sshd_config file:

```
PubkeyAuthentication yes
ChallengeResponseAuthentication yes
AuthorizedKeysFile /etc/ssh/authorized_keys/%u
UsePAM yes
AuthenticationMethods keyboard-interactive:pam,publickey
```

If you are starting from scratch consider using the sshd_config file provided in the samples directory as a starting point. Restart OpenSSH after all changes are made.

PAM

Copy the PAM configuration file for sshd to inject Keeto into the authentication process of OpenSSH:

```
<root>$ cp samples/sshd /etc/pam.d
```

Syslog

Keeto logs to the syslog facility specified in keeto.conf. Adjust your syslog server accordingly. A sample config for syslog-ng can be found in the samples directory that logs Keeto output to a local file.

Directory Service

Keeto consults a Directory Service in order to obtain current access permissions and keys. The relevant entities and their relationship are described in *LDAP Data Model*. General configuration is software dependent and not not outlined here. The samples directory however contains relevant configurations for OpenLDAP and LDIF files for setting up a testing environment.

LDAP Data Model

Keeto obtains all access permissions and the key material from a central Directory Service. This section describes syntax and semantic of the entities involved and how they relate to each other. Certain attributes can be configured in the Keeto configuration file. Those attributes are of the form <x> where x is referring to the key in the Keeto configuration file.

SSH Server

The SSH server entry is the starting point for the determination of access permissions and key material. It specifies the relevant access profiles that shall be taken into account. Keeto locates the right SSH server entry through a unique identifier that is specified in the Keeto configuration file and must match the identifier in the SSH server entry in the Directory Service.

```
objectClass: top
objectClass: keetoSSHServer
```

Attribute	Mandatory	Single-Value	Description
cn	yes	no	RDN of SSH server entry.
uid	yes	no	Unique identifier of SSH server. See also: <ssh_server_uid>.
keetoAccessProfile	yes	no	DN to Keeto access profile.
description	no	no	SSH server description.

Access Profiles

Keeto supports two different access profiles. Direct access profiles provide access to one's own account whereas access on behalf profiles provide access to someone else's account.

Direct Access Profile

A direct access profile specifies a reference to a group of key providers that shall be able to login with it's own account. Each key provider's UID is checked against the UID of the user about to login. If they match the X.509 certificates of the key provider will be taken into account. Optionally keystore options can be specified that are used for all key providers. A direct access profile can be enabled / disabled.

```
objectClass: top
objectClass: keetoAccessProfile
objectClass: keetoDirectAccessProfile
```

Attribute	Mandatory	Single-Value	Description
cn	yes	no	RDN of direct access profile entry.
keetoEnabled	yes	yes	Enable / Disable profile.
keetoKeyProvider	yes	yes	DN to Keeto key provider group.
keetoKeystoreOptions	no	yes	DN to Keeto keystore options.
description	no	no	Direct access profile description.

Access On Behalf Profile

Access on behalf profiles enable authentication on behalf of someone else. For that the UID of the user about to login is searched in the target keystore's. On match all valid keys of all key providers are synced into that target keystore. As with direct access profiles keystore options can be specified optionally and the profile can be enabled / disabled.

```
objectClass: top
objectClass: keetoAccessProfile
objectClass: keetoAccessOnBehalfProfile
```

Attribute	Mandatory	Single-Value	Description
cn	yes	no	RDN of access on behalf profile entry.
keetoEnabled	yes	yes	Enable / Disable profile.
keetoKeyProvider	yes	yes	DN to Keeto key provider group.
keetoTargetKeystore	yes	yes	DN to Keeto target keystore group.
keetoKeystoreOptions	no	yes	DN to Keeto keystore options.
description	no	no	Access on behalf profile description.

Key Provider

A key provider is an entry that provides the key material through X.509 certificate(s). All relevant attributes are specified in the Keeto configuration file to adjust it to different deployments. Key providers are relevant for both direct access profiles and access on behalf profiles.

Attribute	Mandatory	Single-Value	Description
<ldap_key_provider_uid_attr>	yes	no	UID of key provider.
<ldap_key_provider_cert_attr>	yes	no	X.509 certificate of user.

Target Keystore

A target keystore is only relevant for access on behalf profiles. It specifies the accounts that can be used on behalf of the key providers.

Attribute	Mandatory	Single-Value	Description
<ldap_target_keystore_uid_attr>	yes	no	UID of target keystore.

Groups

Both key providers and target keystores are not linked directly to an access profile but rather through a group. For both cases the group member attribute is specified in the Keeto configuration file.

Attribute	Mandatory	Single-Value	Description
<ldap_key_provider_group_member_attr> / <ldap_target_keystore_group_member_attr>	yes	no	DN to key provider / target keystore.

Keystore Options

Keystore options can be optionally specified and linked to access profiles to restrict access with regard to the location a user is connecting from and the space of commands he is allowed to execute.

```
objectClass: top
objectClass: keetoKeystoreOptions
```

Attribute	Mandatory	Single-Value	Description
cn	yes	no	RDN of keystore options entry.
keetoKeystoreOptionFrom	no	yes	authorized_keys 'from' option entry. See also: man sshd.
keetoKeystoreOptionCommand	no	yes	authorized_keys 'command' option entry. See also: man sshd.
description	no	no	Keystore options description.

FAQ

How can I exclude accounts from being processed by Keeto?

Keeto is associated with keyboard-interactive authentication via PAM. Disabling keyboard-interactive authentication for specific users does the trick. This can be configured in the sshd_config file of OpenSSH through the Match directive as follows:

```
Match User foo,bar
    AuthenticationMethods publickey
    AuthorizedKeysFile .ssh/authorized_keys
```