VPN Implementation based on IKEv2 backed by Vault as PKI Engine

Feature Reference

Introduction

Purpose
Provide the new VPN implementation based on IKEv2 rather than using existing L2TP implementation.

Document History

v1 - Khosrow Moossavi - April 2nd 2018

Feature Specifications

- provide a global settings to switch between L2TP and IKEv2 (only one can be active throughout an installation)
- out of the box configuration of ipsec will be provided in /etc/ipsec.d/ikev2.conf
- authentication will be done with EAP and Public Key
- it will use self-signed certificates per domain act as CA on VRs
- PKI backend engine (out of the box support)
  - Vault by HashiCorp
  - Default internal implementation (TBD on dev@)
  - External Services (such as Let's Encrypt) (TBD on dev@)
  - Bring Your Own Certificate (BYOC) model (TBD on dev@)
- explain configuration characteristics:
  - configuration parameters or files introduced/changed
  - branding parameters or files introduced/changed
  - highlight parameters for performance tweaking
  - highlight how installation/upgrade scenarios change
- deployment requirements (fresh install vs. upgrade) if any
- system requirements: no special requirements needed
- interoperability and compatibility requirements:
  - Tested on Debian 7 on VR
  - All major OSes (Linux, Windows 10, Mac X) as client of VPN
- list localization and internationalization specifications
  - one language key added (message.enabledvpn.ca.certificate)
- explain the impact and possible upgrade/migration solution introduced by the feature
- explain levels or types of users communities of this feature (e.g. admin, user, etc)
  - this will be used by both infra admin, customer admin and customer user as it will be the Remove Access VPN implementation

Architecture and Design description

- the main feature is pretty straight forward, a global setting is added to distinguish between L2TP and IKEv2 implementation
- the scripts changes on VR are pretty straight forward as well, the type of VPN added in the command being sent to VR and the corresponding ipsec config file will be loaded
- the main part of the design document will be around using external or start implementing internal PKI backend engine. which we will have multiple options (at least two)
  - Using Vault as the PKI engine (recommended by author and fully implemented as of Apr 2018)
  - Using Cloudstack as a self contained PKI engine (it's not recommended and it's not implemented)
  - Using external services (such as Let's Encrypt) to generate and sign certificates (this is nice to have but will need to be discussed on ML)
  - Bring Your Own Certificate (BYOC) model (this is nice to have but will need to be discussed on ML)
- list of added settings are

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pki.engine.certificate.brand</td>
<td>Brand name to be used in Certificate's common name</td>
<td>Cloudstack</td>
</tr>
<tr>
<td>pki.engine.certificate.common.name</td>
<td>Certificate's common name template (brand will be filled from 'pki.engine.certificate.brand', domain will be provided on the fly</td>
<td>_BRAND__VPN__DOMAIN__CA</td>
</tr>
<tr>
<td>pki.engine.vault.cca.ttl</td>
<td>Vault PKI root CA TTL (e.g. 87600h)</td>
<td>87600h</td>
</tr>
<tr>
<td>pki.engine.vault.enabled</td>
<td>Enable Vault as the backend PKI engine</td>
<td>false</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>pki.engine.vault.mount.path</td>
<td>Vault PKI mount point prefix (must not end with trailing slash)</td>
<td>pki/cloudstack</td>
</tr>
<tr>
<td>pki.engine.vault.role.name</td>
<td>Vault PKI role name</td>
<td>cloudstack-vpn</td>
</tr>
<tr>
<td>pki.engine.vault.role.ttl</td>
<td>Vault PKI role TTL (e.g. 43800h)</td>
<td>43800h</td>
</tr>
<tr>
<td>pki.engine.vault.token</td>
<td>Token to access Vault</td>
<td>(empty)</td>
</tr>
<tr>
<td>pki.engine.vault.token.role.id</td>
<td>App Role id to be used to fetch token to access Vault</td>
<td>(empty)</td>
</tr>
<tr>
<td>pki.engine.vault.token.secret.id</td>
<td>Secret id to be used to fetch token to access Vault</td>
<td>(empty)</td>
</tr>
<tr>
<td>pki.engine.vault.ttl</td>
<td>Vault PKI TTL (e.g. 87600h)</td>
<td>87600h</td>
</tr>
<tr>
<td>pki.engine.vault.url</td>
<td>Full URL of Vault endpoint (e.g. <a href="http://127.0.0.1:8200">http://127.0.0.1:8200</a>)</td>
<td><a href="http://127.0.0.1:8200">http://127.0.0.1:8200</a></td>
</tr>
</tbody>
</table>

**Web Services APIs**

*list changes to existing web services APIs and new APIs introduced with signatures and throughout documentation*

- **Added API**
  - `ListVpnCaCertificateCmd`
    - **input:**
      - domain (uuid)
    - **output (CertificateResponse):**
      - certificate: The client certificate
      - privateKey: Private key for the certificate
      - caCertificate: The CA certificate(s)
- **Modified API**
  - `RemoteAccessVpnResponse`: two additional fields
    - type: the type of remote access vpn implementation (e.g. l2tp or ikev2)
    - certificate: the client certificate

**UI flow**

- the only changes on the UI are the fact that we don’t have Preshared Key anymore rather we will have Certificates (and user should be able to download them)

**IP Clearance**

- [https://github.com/BetterCloud/vault-java-driver](https://github.com/BetterCloud/vault-java-driver) **com.bettercloud.vault-java-driver** (v3.1.0 for now, I might need to upgrade it to v3.1.1 soon based on an open PR I have)

**Usage Impact**

- None