Building an safe access into your cloud app with HashiCorp Vault

FOSDEM - 03/02/2018
Who Am I

- Thomas Gerbet (@el_suisse)
- @TuleapOpenALM contributor and maintainer
- @GreHackConf organizer
- Alpaca lover
A Bit of Context

- **SaaS provider**
- **Main contributors of the software**
  - We can develop specific feature for the SaaS usage
- **Web based**
  - REST API
  - PHP
- **Single Tenant**
- **Support**
SaaS and Access Management: Challenges

● **Customer’s data are only as safe as the provider is**
  - Do not maintain unnecessary or permanent accesses
  - Restrict access to selected team members
  - Something bad will happen someday, plan for it

● **Accountability / Auditability**
  - What?
  - When?
  - Who?

● **Keep things usable**
  - Must work for people doing the day-to-day job
HashiCorp Vault
# HashiCorp Vault Backends

<table>
<thead>
<tr>
<th>Authentication</th>
<th>Audit</th>
<th>Secrets</th>
</tr>
</thead>
<tbody>
<tr>
<td>- AppRole</td>
<td>- File</td>
<td>- AWS</td>
</tr>
<tr>
<td>- AWS</td>
<td>- Syslog</td>
<td>- Consul</td>
</tr>
<tr>
<td>- Google Cloud</td>
<td>- Socket</td>
<td>- Databases</td>
</tr>
<tr>
<td>- Kubernetes</td>
<td></td>
<td>- Key/Value</td>
</tr>
<tr>
<td>- GitHub</td>
<td></td>
<td>- Nomad</td>
</tr>
<tr>
<td>- LDAP</td>
<td></td>
<td>- PKI</td>
</tr>
<tr>
<td>- MFA</td>
<td></td>
<td>- RabbitMQ</td>
</tr>
<tr>
<td>- Okta</td>
<td></td>
<td>- SSH</td>
</tr>
<tr>
<td>- RADIUS</td>
<td></td>
<td>- TOTP</td>
</tr>
<tr>
<td>- TLS certs</td>
<td></td>
<td>- Transit</td>
</tr>
<tr>
<td>- Tokens</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Username/Password</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Building its own dynamic Vault secret backend

- **Vault supports plugins since August 2017**
  - Basic knowledge of Go is enough
  - You can build for your specific use case
  - Still get all the nice Vault features

- **Support is needed in your software**
  - Be able to create (and revoke) short lived accounts
  - Authenticate requests coming from Vault
Public-Key Cryptography is Awesome

- **No hardcoded credentials**
  - Vault generates and stores the private key
  - Instances of your app only knows the public key to authenticate requests

- **libsodium (Ed25519 signatures)**
  - Modern cryptography
  - Bindings widely available
    - Go
      - [golang.org/x/crypto/ed25519](http://golang.org/x/crypto/ed25519)
    - PHP
      - ≥ 7.2: standard library
      - ≤ 7.1: extension or polyfill (thanks @ParagonIE)
Requesting an account

Team member → Vault

Authenticate

Token

Get account(Token, FQDN)

Create account(Username, Password, Expiration, Signature)

Ok!

Username, Password

Authenticate(Username, Password)

Team member → Vault → myTuleap instance

Team member → Vault → myTuleap instance
Plan for the worst

- **Revocation**
  - Immediate
  - Granularity:
    - One specific lease
    - All leases of a specific user
    - All leases

- **Seal the Vault**
  - All operations are blocked
  - Lets you minimize and assess damage in case of a detected intrusion
Outcome

- **HashiCorp Vault**
  - Integrates nicely in your existing infrastructure
  - Highly flexible secrets management
  - Audit capabilities

- **One more sensitive endpoint in our software 😞**
  - Still better than hardcoded credentials though

- **Usability**
  - Authenticate against Vault → Request account → Log into the instance
  - Only CLI 😊
Questions?