Secrets Tyeps 参考

Barbican中的每一个机密都有类型。机密的类型被用来说明存储在Barbican中的不同类型的机密数据。特定的机密类型通过机密元数据中的secret_type属性列出。
可能用到的机密类型：

- symmetric - 用来存储字节数据，例如对称加密的密钥
- public - 用来存储非对称密钥对的公钥
- private - 用来存储非对称密钥对的私钥
- passphrase - 用来存储纯文本密码
- certificate - 用来存储加密证书，例如X.509证书
- opaque - 用于兼容之前版本API不带类型的机密。新版推荐指定一种机密类型

Symmetric

symmetric机密类型用于存储字节数组，如对称加密的密钥。与symmetric类型一起使用的content_type是application/octet-stream。当使用单个POST请求存储对称密钥时，必须对数据进行编码，以便被包括在JSON请求里。这种情形下，可以使用base64对对称密钥进行编码。

示例 1.1

创建一个用于AES-256-CBC加密的对称密钥并存储到Barbican中。首先，我们将看到如何使用 curl 通过单次POST请求创建。

```
# Create an encryption_key file with 256 bits of random data
dd bs=32 count=1 if=/dev/urandom of=encryption_key

# Encode the contents of the encryption key using base64 encoding
KEY_BASE64=$(base64 < encryption_key)

# Send a request to store the key in Barbican
curl -vv -H "X-Auth-Token: $TOKEN" -H 'Accept: application/json' \
-H 'Content-Type: application/json' \
-d '{"name": "AES encryption key",
     "secret_type": "symmetric",
     "payload": "'"$KEY_BASE64"'",
     "payload_content_type": "application/octet-stream",
     "payload_content_encoding": "base64",
     "algorithm": "AES",
     "bit_length": 256,
     "mode": "CBC"}' \
http://localhost:9311/v1/secrets | python -m json.tool
```

此时应该返回被创建的机密的引用：

```
{
  "secret_ref": "http://localhost:9311/v1/secrets/48d24158-b4b4-45b8-9669-
d9f0ef793c23"
}
```

我们可以通过上面的引用来检索该机密的元数据：

```
curl -vv -H "X-Auth-Token: $TOKEN" -H 'Accept: application/json' \
http://localhost:9311/v1/secrets/48d24158-b4b4-45b8-9669-d9f0ef793c23 |
python -m json.tool
```

元数据应当列出该symmetric机密的可用信息：

```
{
    "algorithm": "AES",
    "bit_length": 256,
    "content_types": {
        "default": "application/octet-stream"
    },
    "created": "2015-04-08T06:24:16.600393",
    "creator_id": "3a7e3d2421384f56a8fb6cf082a8efab",
    "expiration": null,
    "mode": "CBC",
    "name": "AES encryption key",
    "secret_ref": "http://localhost:9311/v1/secrets/48d24158-b4b4-45b8-
9669-d9f0ef793c23",
    "secret_type": "symmetric",
    "status": "ACTIVE",
    "updated": "2015-04-08T06:24:16.614204"
}
```

content_types属性说明的内容类型可以用来检索payload。在这个示例中，使用的是默认的类型：application/octet-stream。我们可以用它来检索payload：

```
# Retrieve the payload and save it to a file
curl -vv -H "X-Auth-Token: $TOKEN" \
-H 'Accept: application/octet-stream' \
-o retrieved_key \
http://localhost:9311/v1/secrets/48d24158-b4b4-45b8-9669-
d9f0ef793c23/payload
```

该retrieved_key文件中包含了我们开始时使用的字节数组。注意，Barbican是以二进制的形式返回的字节数据，而不是base64.这是因为payload_content_encoding只在向Barbican提交密钥时使用。

Public

此机密类型常用来存储非对称密钥对的公钥，例如可用于存储RSA密钥对的公钥。目前，public机密仅支持一种文件格式：SubjectPublicKeyInfo，由X.509 RFC 5280定义的DER编码结构，使用PEM头和尾进行base64编码。这种类型的公钥一般是通过OpenSSL工具生成。用于public机密的内容类型是application/octet-stream.当通过单个POST请求存储public机密时，因为JSON不接受换行符，所以文件内容必须编码。这种情况下，文件内容必须是base64编码过的，并且base64可以使用内容编码。

示例 2.1

生成RSA密钥对并将公钥存入Barbican。在这个例子中，我们将使用POST只发送元数据，然后使用PUT发送payload。

```
# Create the RSA keypair
openssl genrsa -out private.pem 2048

# Extract the public key
openssl rsa -in private.pem -out public.pem -pubout

# Submit a metadata-only POST
curl -vv -H "X-Auth-Token: $TOKEN" \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
-d '{"name": "RSA Public Key",
     "secret_type": "public",
     "algorithm": "RSA"}' \
http://localhost:9311/v1/secrets | python -m json.tool
```

此时应该返回被创建的机密的引用：

```
200 OK

{
  "secret_ref": "http://localhost:9311/v1/secrets/cd20d134-c229-417a-a753-
86432ad13bad"
}
```

接着我们通过PUT请求使用这个引用将payload添加到机密中：

```
curl -vv -X PUT -H "X-Auth-Token: $TOKEN" \
-H 'Accept: application/json' \
-H 'Content-Type: application/octet-stream' \
--data-binary @public.pem \
http://localhost:9311/v1/secrets/cd20d134-c229-417a-a753-86432ad13bad
```

服务器此时会以2xx响应标识PUT请求成功执行：

```
204 - No Content
```

现在我们可以请求元数据，并列出新的content-type：

```
curl -vv -H "X-Auth-Token: $TOKEN" \
-H 'Accept: application/json' \
http://localhost:9311/v1/secrets/cd20d134-c229-417a-a753-86432ad13bad |
python -m json.tool
```

```
{
    "algorithm": "RSA",
    "bit_length": null,
    "content_types": {
        "default": "application/octet-stream"
    },
    "created": "2015-04-08T21:45:59.239976",
    "creator_id": "3a7e3d2421384f56a8fb6cf082a8efab",
    "expiration": null,
    "mode": null,
    "name": "RSA Public Key",
    "secret_ref": "http://localhost:9311/v1/secrets/cd20d134-c229-417a-
a753-86432ad13bad",
    "secret_type": "public",
    "status": "ACTIVE",
    "updated": "2015-04-08T21:52:57.523969"
}
```

最后，我们可以通过content_type列出的内容类型来获取公钥：

```
curl -vv -H "X-Auth-Token: $TOKEN" \
-H 'Accept: application/octet-stream' \
-o retrieved_public.pem \
http://localhost:9311/v1/secrets/cd20d134-c229-417a-a753-
86432ad13bad/payload
```

retrieved_public.pem文件中保存着跟我们最开始使用的public.pem文件中一样的内容。

示例 2.2

创建RSA密钥对并将公钥保存到Barbican中。在这个示例中，我们使用单个POST请求。

```
# Create the RSA keypair
openssl genrsa -out private.pem 2048

# Extract the public key
openssl rsa -in private.pem -out public.pem -pubout

# Base64 encode the contents of the public key
PUB_BASE64=$(base64 < public.pem)

curl -vv -H "X-Auth-Token: $TOKEN" \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
-d '{"name": "RSA Public Key",
    "secret_type": "public",
    "payload": "'"$PUB_BASE64"'",
    "payload_content_type": "application/octet-stream",
    "payload_content_encoding": "base64",
    "algorithm": "RSA"}' \
http://localhost:9311/v1/secrets | python -m json.tool
```

此时应该会返回创建的机密的引用：

```
200 OK

{
  "secret_ref": "http://localhost:9311/v1/secrets/d553f0ac-c79d-43b4-b165-
32594b612ad4"
}
```