Login mit signierten ssh-Schlüsseln

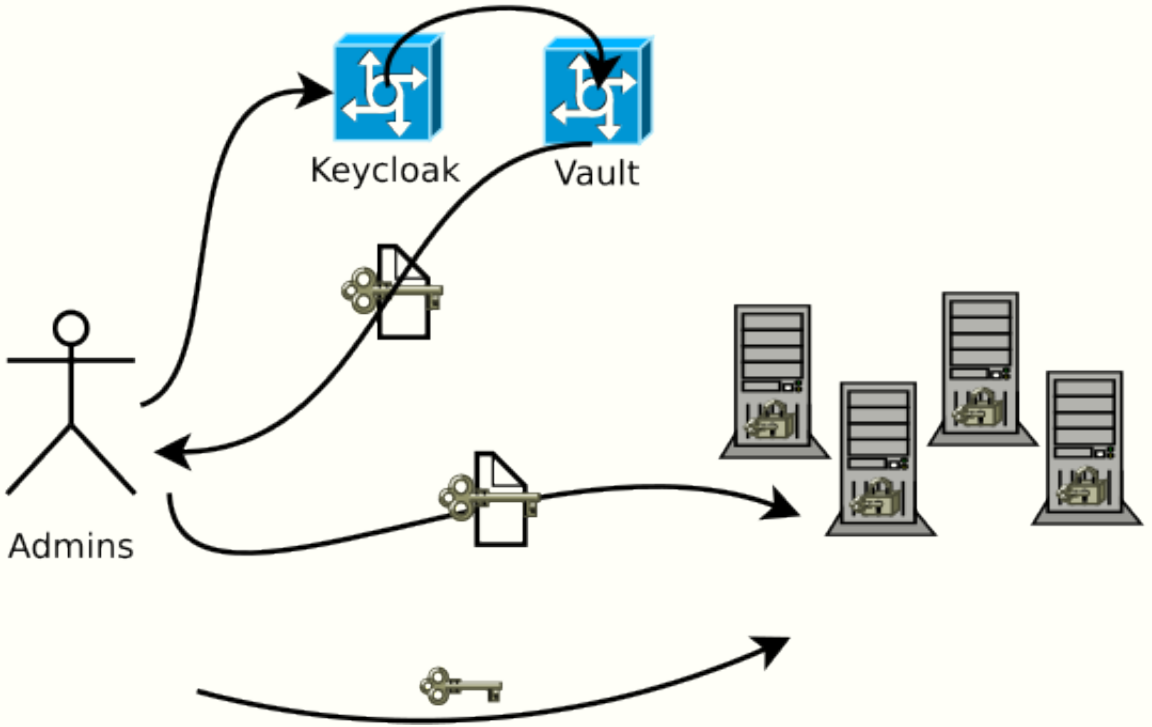
Christopher J. Ruwe <cjr@cyberiada.de>

Cyberiada GmbH

Problem

- ohne großen Aufwand sollen Personen zum Login auf eine (große) Menge an Hosts berechtigt werden
- Rechte sollen einfach und schnell entzogen werden können
- es soll keine aufwendige Authn/Authz-Infra (AD/krb) gepflegt werden

Lösungs-Angebot



Lösungs-Angebot

```
Type: ssh-rsa-cert-v01@openssh.com user certificate
Public key: RSA-CERT SHA256:tJYvjQig65sxMxlcoH9MJyboWu2Gru092ORNfh1XhNs
Signing CA: RSA SHA256:tZIipsTz7DJE6kGbESP5d+sNvgryqK3fWPiIrDvzhFk (using rsa-sha2-256)
Key ID: "vault-oidc-belphegor@he.ll<long_number>"
Serial: 16626502850102714285

Valid: from 2020-11-14T17:41:51 to 2020-12-16T17:42:21
Principals:
    hal9000
Critical Options: (none)
Extensions:
    permit-X11-forwarding
    permit-agent-forwarding
    permit-port-forwarding
    permit-pty
    permit-user-rc
```

wie hilft das?

- Schlüssel-Signatur nur gegegen Token von zentralem authn/authz Mechanismus
- Gültigkeit kurz
- Schlüssel ungültig heißt Login-Rechte weg

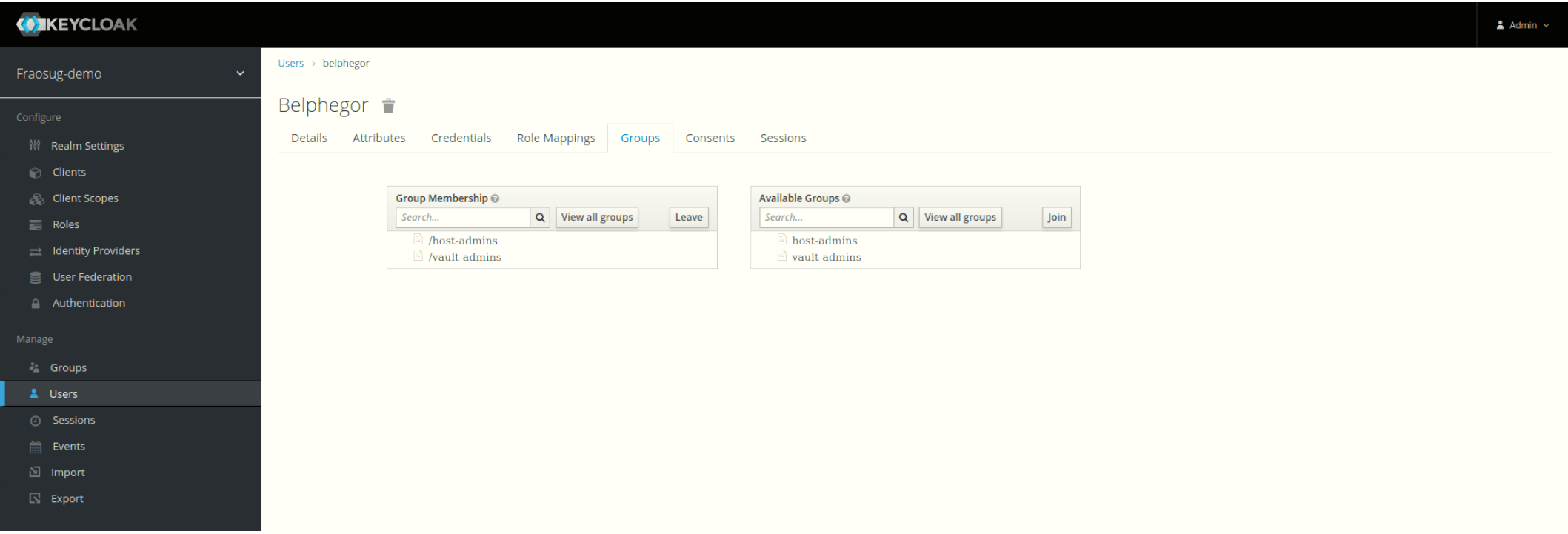
wie läuft das?

- keine persönlichen Nutzer mehr
- nur noch Funktionsnutzer mit entsprechenden Rechten
- Zuordnung über Audit-Logs

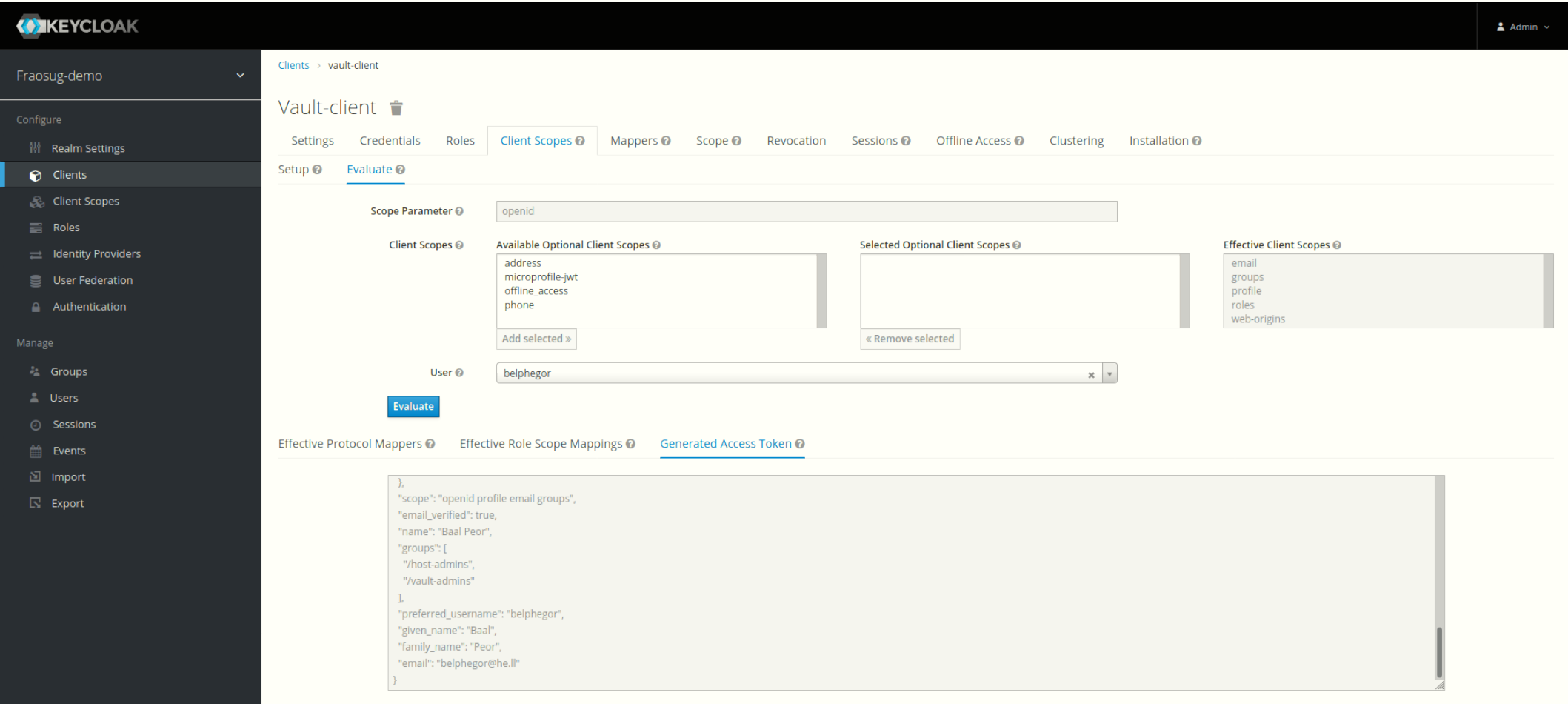
Mechanismus

- Nutzer-Datenbank entweder direkt oder *federated* in Keycloak
- dort Pflege von Gruppen-Zuordnung
- Übermittlung im OIDC-Token

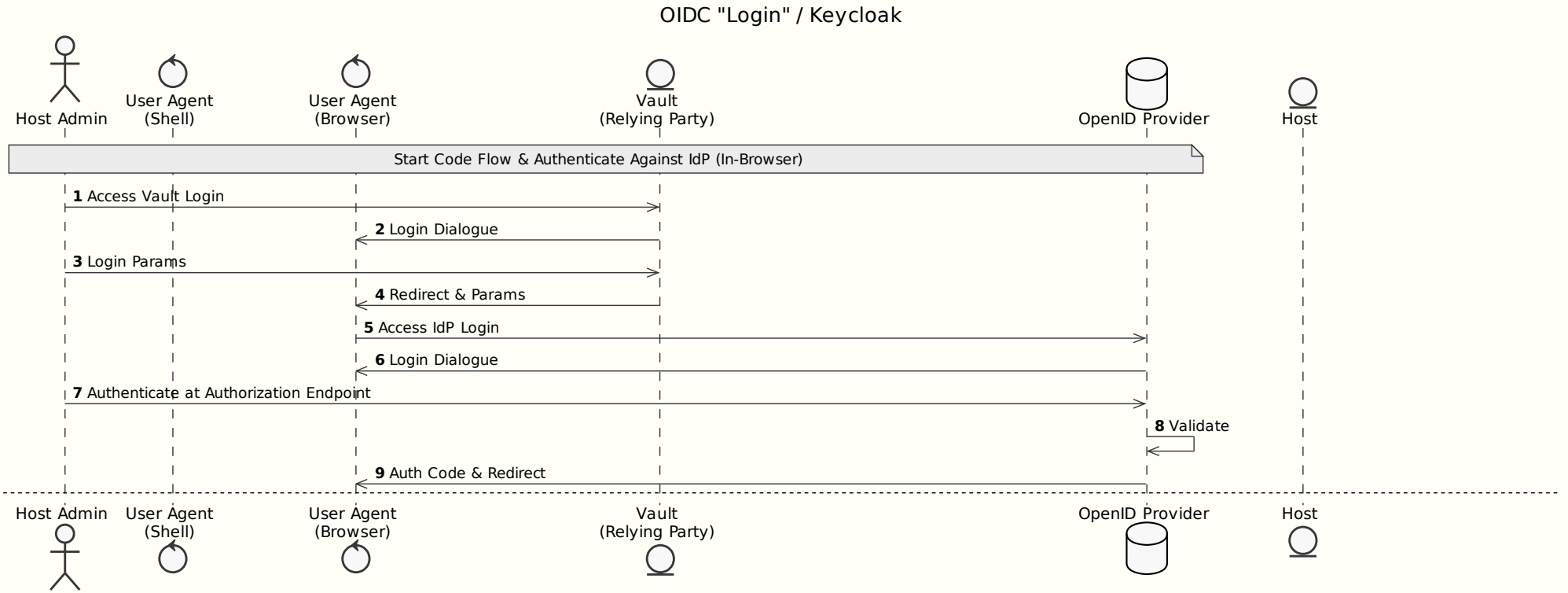
Identität und Rechte-Verwaltung



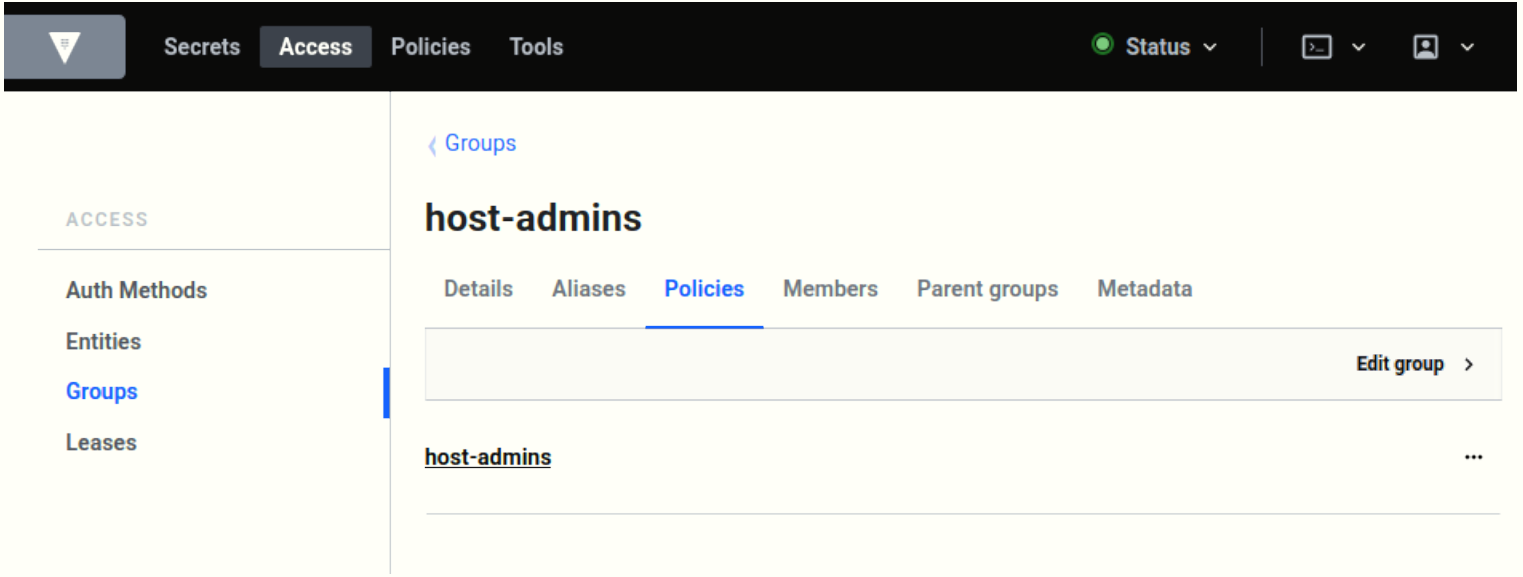
Token-Mapping



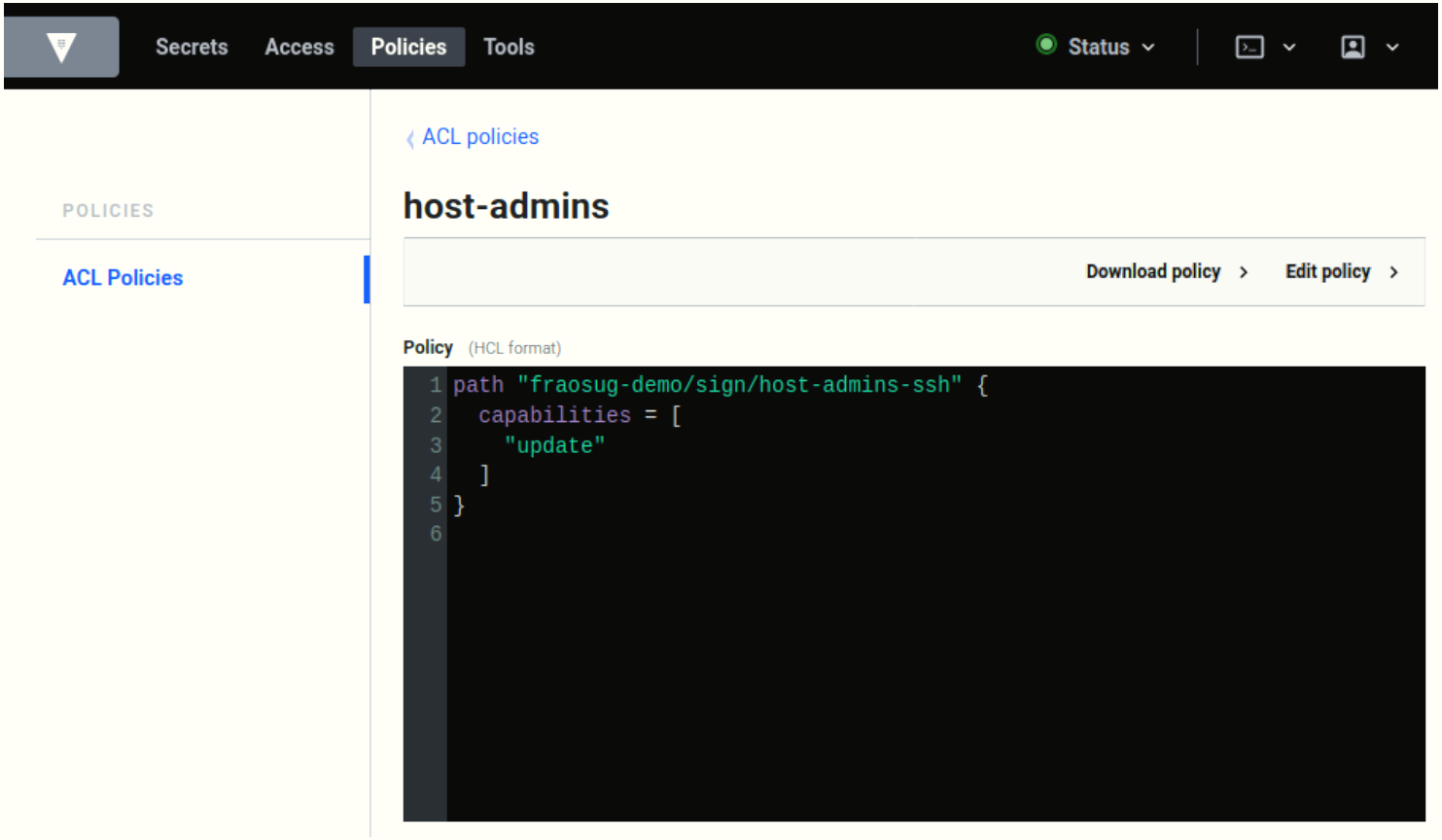
Verfahren


OIDC "Login" / Keycloak
Host Admin
User Agent (Shell)
User Agent (Browser)
Vault (Relying Party)
OpenID Provider
Host
Start Code Flow & Authenticate Against IdP (In-Browser)
1 Access Vault Login
2 Login Dialogue
3 Login Params
4 Redirect & Params
5 Access IdP Login
6 Login Dialogue
7 Authenticate at Authorization Endpoint
8 Validate
9 Auth Code & Redirect
Host Admin
User Agent (Shell)
User Agent (Browser)
Vault (Relying Party)
OpenID Provider
Host

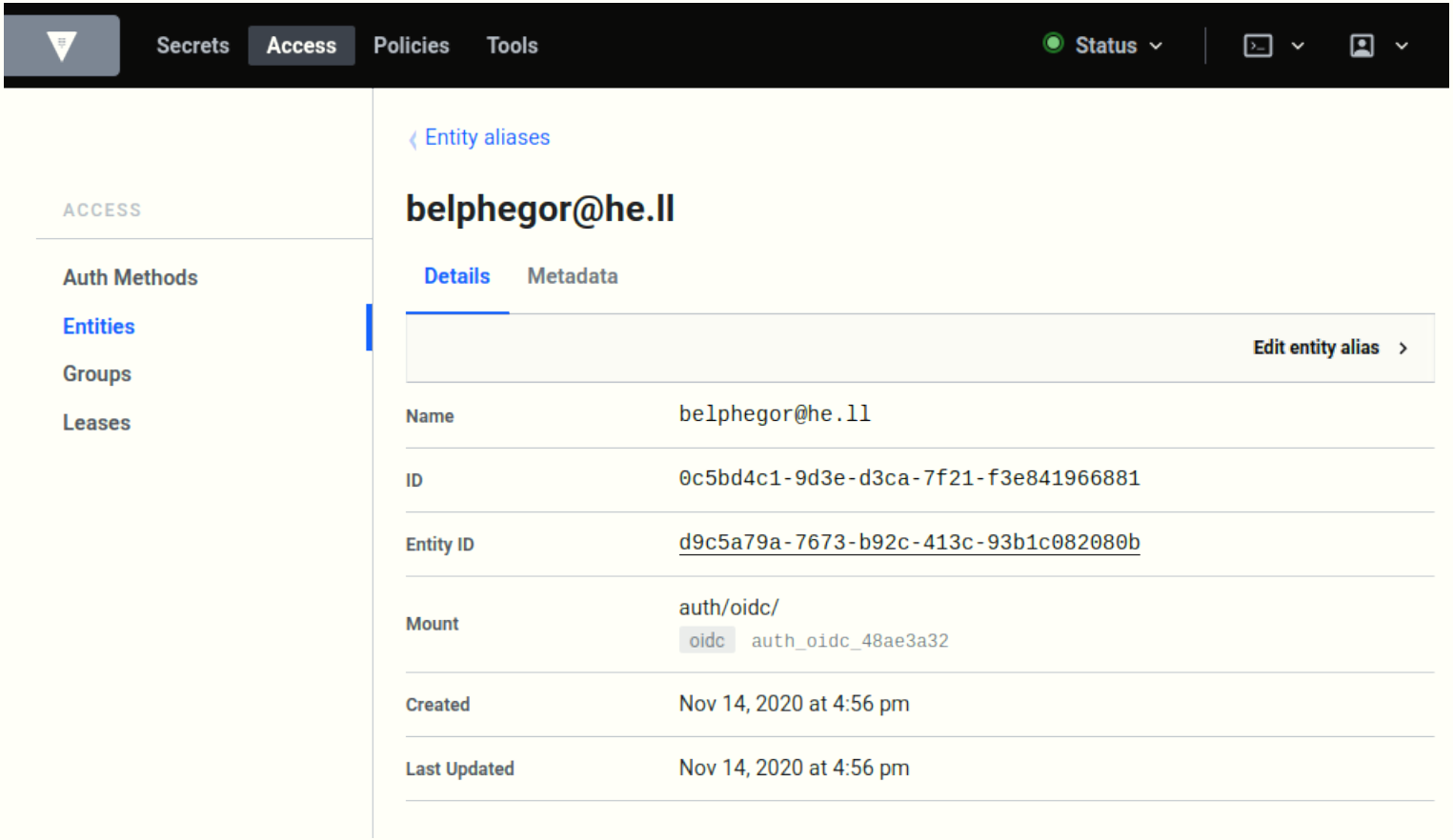
Gruppen-Zuordnung



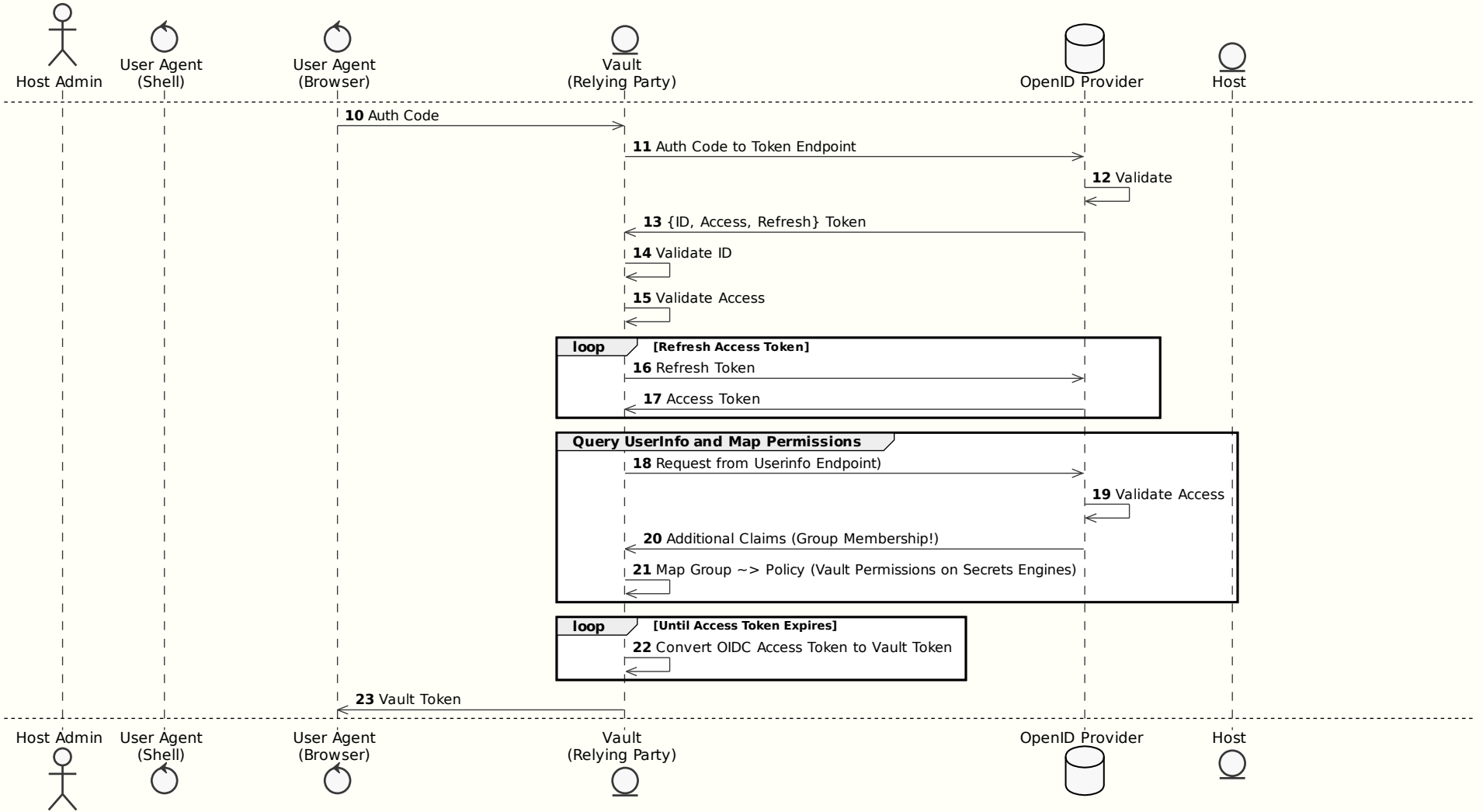
Rechte-Vergabe per Policy



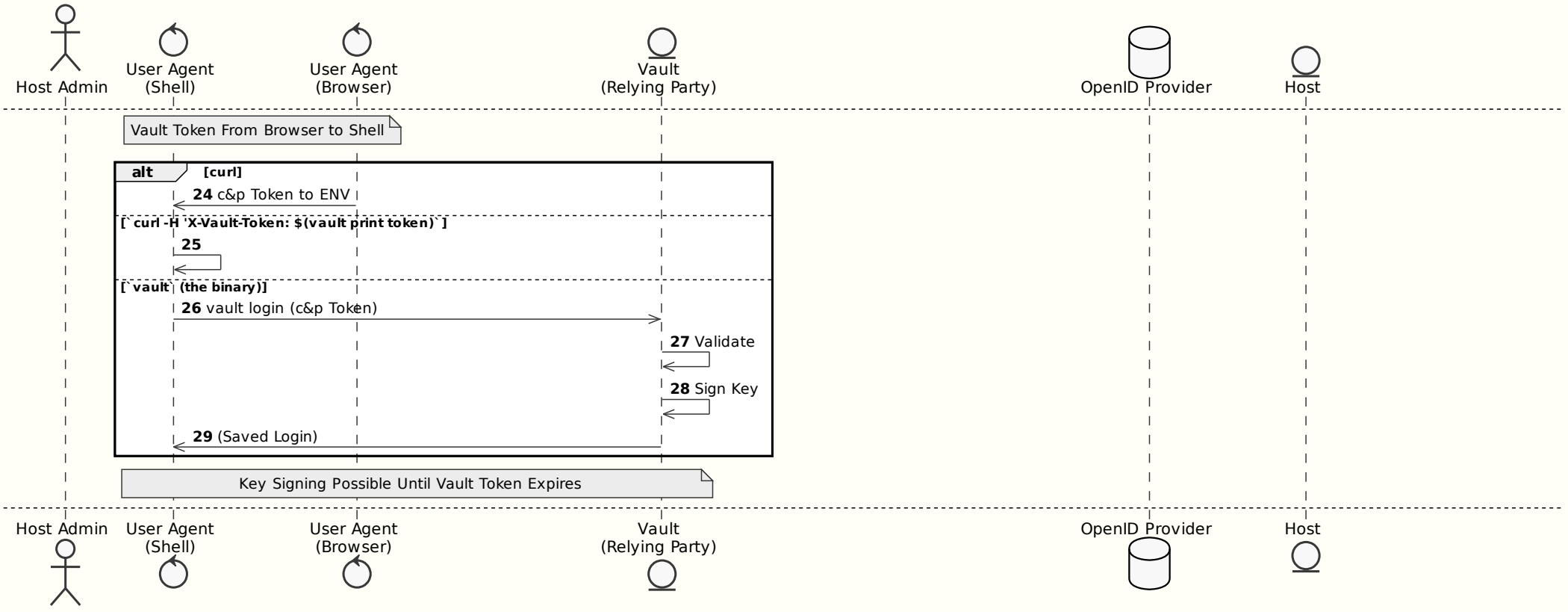
Transport per OIDC



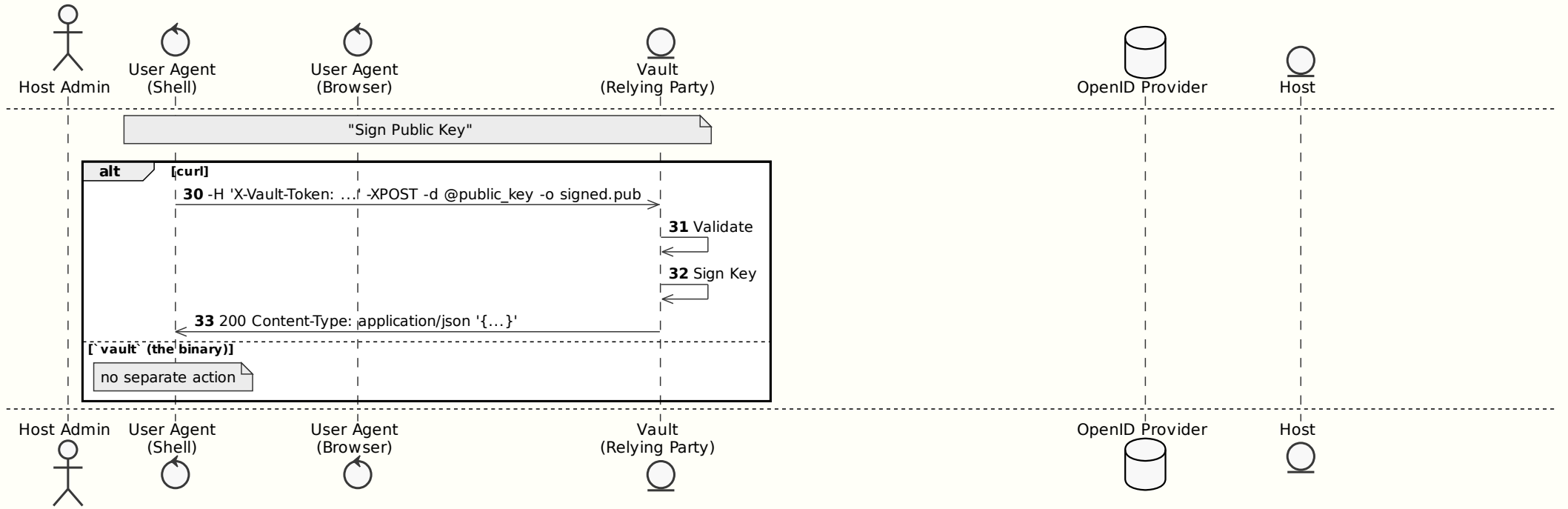
Transport Authentication & UserInfo to Vault



Use Vault Token



How to Sign Keys



Rechte-*Übersetzung* per Rolle

Signatur eines Schlüssels

```
{
curl \
  -X PUT \
  -H "X-Vault-Request: true" \
  -H "X-Vault-Token: $(vault print token)" \
  -d@- \
  http://192.168.99.67:8200/v1/fraosug-demo/sign/host-admins-ssh \
<<EOT
{
  "cert_type":"user",
  "extensions": {
    "permit-X11-forwarding":"",
    "permit-agent-forwarding":"",
    "permit-port-forwarding":"",
    "permit-pty":"",
    "permit-user-rc":""
  },
  "public_key":"$(cat /home/cjr/media/src/50-edu/50-talks/201117-cjr--fraosug-ssh/doc/fraosug.pub)",
  "valid_principals":"hal9000"
}
EOT
} |  jq \
   -r '.data.signed_key' \
|  ssh-keygen -f - -L
```

Login

```
ssh  \
  -i .../fraosug \
  -oCertificateFile=.../fraosug-signed.pub  \
  hal9000@192.168.99.68

vault ssh \
  -mode=ca \
  -mount-point=fraosug-demo \
  -role=host-admins-ssh \
  hal9000@192.168.99.68
```

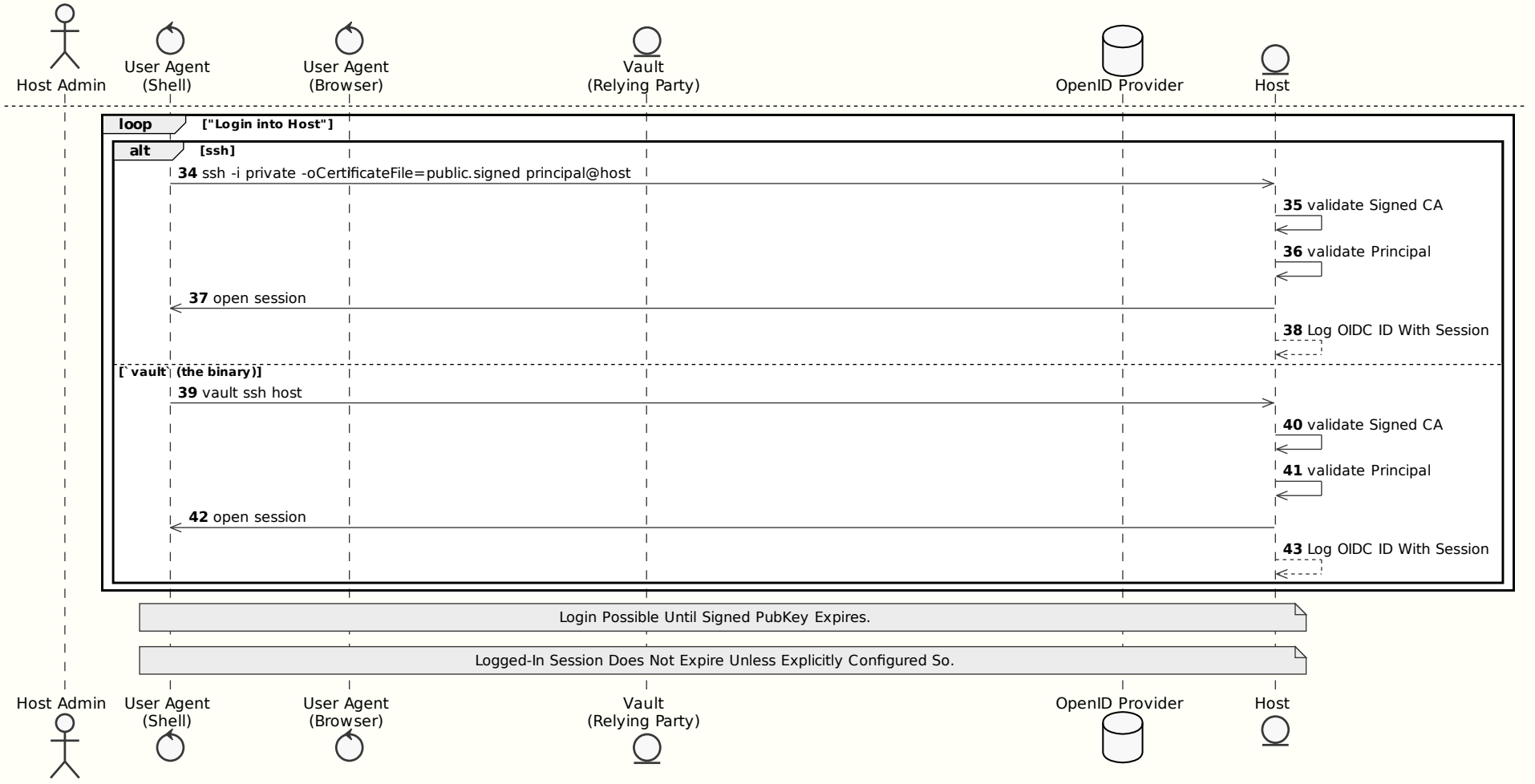
Audit-Trail (1)

```
Nov 14 17:12:37 moon sshd[5716]: Postponed publickey for hal9000 from
192.168.99.1 port 40154 ssh2 [preauth]
Nov 14 17:12:37 moon sshd[5716]: Accepted publickey for hal9000 from
192.168.99.1 port 40154 ssh2: RSA-CERT
SHA256:TGvFsbj58aEX1hP3RMSUT/7t4y5lorasgfr9xhLYZwg ID vault-oidc-
belphegor@he.ll-4c6bc5b1b8f9f1a117d613f744c4944ffeede32e65a2b6ac81fafdc612d86708
(serial 67054684129554624 74) CA RSA
SHA256:rW3eVJ1WXeCc/rou0g3MXIJXHlOYcJdf10/nYDWX250
Nov 14 17:12:37 moon sshd[5716]: pam_unix(sshd:session): session opened for user
hal9000 by (uid=0)
Nov 14 17:12:37 moon systemd-logind[706]: New session 31 of user hal9000.
```

Audit-Trail (2)

```
type=USER_LOGIN msg=audit(1605373958.051:430): pid=5716 uid=0 auid=1000 ses=31
msg='op=login id=1000 exe="/usr/sbin/sshd" hostname=192.168.99.1
addr=192.168.99.1 terminal=/dev/pts/1 res=success'
type=USER_ACCT msg=audit(1605374084.981:439): pid=5827 uid=1000 auid=1000 ses=31
msg='op=PAM:accounting grantors=pam_permit acct="hal9000" exe="/usr/bin/sudo"
hostname=? addr=? terminal=/dev/pts/1 res=success'
type=USER_CMD msg=audit(1605374084.981:440): pid=5827 uid=1000 auid=1000 ses=31
msg='cwd="/home/hal9000" cmd="ls" terminal=pts/1 res=success'
type=CRED_REFR msg=audit(1605374084.981:441): pid=5827 uid=0 auid=1000 ses=31
msg='op=PAM:setcred grantors=pam_permit,pam_cap acct="root" exe="/usr/bin/sudo"
hostname=? addr=? terminal=/dev/pts/1 res=success'
```

Logging in to Hosts



Fallstricke

- es handelt sich beim signierten Schlüssel konzeptionell um ein Zertifikat, **kein Token**
- auch wenn das Zertifikat per `` `-i ...` `` übergeben werden kann, ist es kein neues Secret

Exkurs: terraform

- API-Möhre

- *recte*: Erzeugung und Konfiguration von Ressourcen über eine API
- ursprünglich Cloud-Ressourcen, mittlerweile alles mögliche inkl. Pizza-Bestellung

Exkurs: terraform

```
terraform {
  required_providers {
    keycloak = {
      source = "mrparkers/keycloak"
      version = "2.0.0"
    }

    vault = {
      source = "hashicorp/vault"
      version = "2.15.0"
    }
  }
  backend "pg" {}
}
```

```
provider "keycloak" {
  client_id = "admin-cli"
  username = "terraform"
  password = "..."
  url = var.endpoint-keycloak
}

provider "vault" {
  address = var.endpoint-vault
  skip_tls_verify = true
  token = "...."
}
```

Exkurs: terraform

```
{
    "version": 3,
    "serial": 1,
    "lineage": "4fc1a10d-9a41-e161-353a-06418f6d2f59",
    "backend": {
        "type": "pg",
        "config": {
            "conn_str": "postgres://terraform:imustshootyouafteryouhavereadthis-<...>@192.168.99.66/terraform",
            "schema_name": null,
            "skip_schema_creation": null
        },
        "hash": 241288904
    },
    "modules": [ <...> ]
}
```