HashiCorp Vault

Vault Governance

Vault Onboarding Program

A 7 week guided community environment
Assisting customers with onboarding and adoption

Vault Architecture Deep dive

- Kick Off & Architectural Quickstart
- Architectural Deep-dive
- Migrating from Vault OSS*

Identity and Access Management

- Namespaces, ACL Policies, Auth Methods
- Community Office Hours
- Using Vault with Kubernetes*

Operationalize

- Telemetry & Monitoring, DR Operations, Runbooks

Secrets Management

- How to consume Vault into applications
- Vault Dynamic Secrets*

Ready for Production!

- Vault Governance
- Exit Ramp & Operational Readiness Checklist (asynchronous)

Agenda

Sentinel	01
Control Groups	02
Quotas	03

01

Sentinel

Overview

- Sentinel is an embeddable ‘policy as code’ framework to enable fine-grained, logic-based policy decisions that can be extended to source external information to make decisions
- Sentinel policies are used in combination with ACL policies and policy templates

Working with Sentinel policies

- Vault injects data into the Sentinel runtime environment, including properties for: requests, replication, tokens, Identity secrets engine, MFA, and Control Groups
- Sentinel Properties List
- Sentinel policies are written in a domain specific language
- Manage policies via HTTP API, CLI, or web UI
- The root token or tokens with the **root** policy attached are **exempt** from Sentinel policies!

Sentinel policy structure

```
import "<library>"
<variable> = <value>

<name> = rule { <condition_to_evaluate> }

main = rule {

      <condition_to_evaluate>

}
```

Example Sentinel policy

```
import "sockaddr"
import "strings"
# Only evaluated for update operations against transit/ path
precond = rule {
    request.operation in [ "update" ] and
    strings.has_prefix(request.path, "transit/")
}
# Requests must originate from our private IP range
cidrcheck = rule {
    sockaddr.is_contained(request.connection.remote_addr,
"122.22.3.4/32")
}
# Check the precondition before executing the cidrcheck
main = rule when precond {
    cidrcheck
}
```

Policy types

- Sentinel allows you to write complex logic and use external information like client CIDR
- Two types **Endpoint Governing Policy** (EGP) & **Role Governing Policy** (RGP)
  - EGPs are applied to particular paths
  - RGPs are applied to tokens, Identity entities, or Identity groups
- Enforcement levels: **advisory**, **soft-mandatory**, or **hard-mandatory**

Endpoint Governing Policies (EGPs)

- API: /sys/policies/egp/
- EGPs are tied to particular paths
- Access to as much information in the request as possible
- Can be tied to all authenticated and most unauthenticated paths
- Denote suffix with glob character (*****) for example: **my-secret-path/***
- Path of just * affects all authenticated and login requests

EGP example

‘Break Glass’ policy denies access when token created prior to specified time

```
import "time"


main = rule when not request.unauthenticated {

  time.load(token.creation_time).unix >

  time.load("2020-01-015T07:25:00Z").unix

}
```

Role Governing Policies (RGPs)

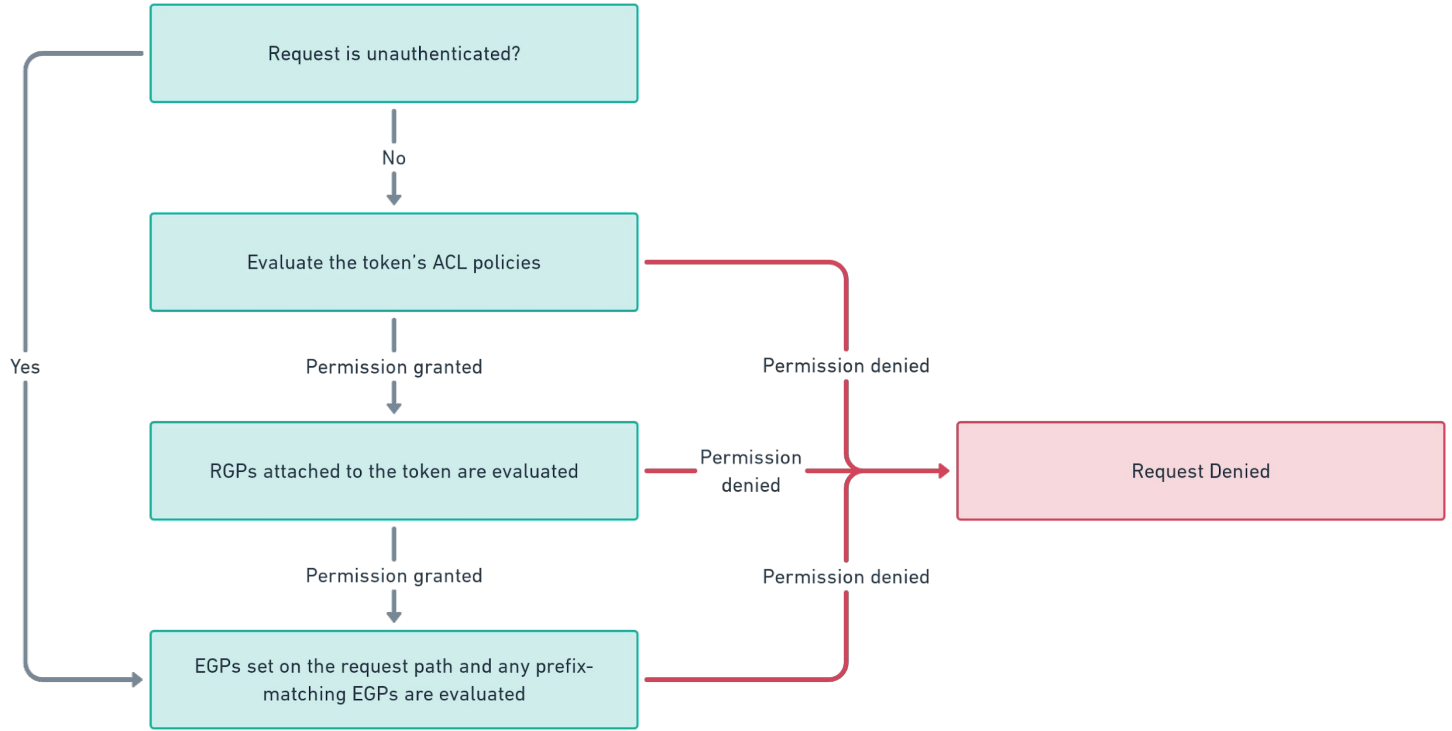
- API: /sys/policies/rgp/
- RGPs are tied to tokens, Identity entities, or Identity groups
- Access to a rich set of controls across many aspects of Vault

RGP example

Use available Identity secrets engine properties to make decisions

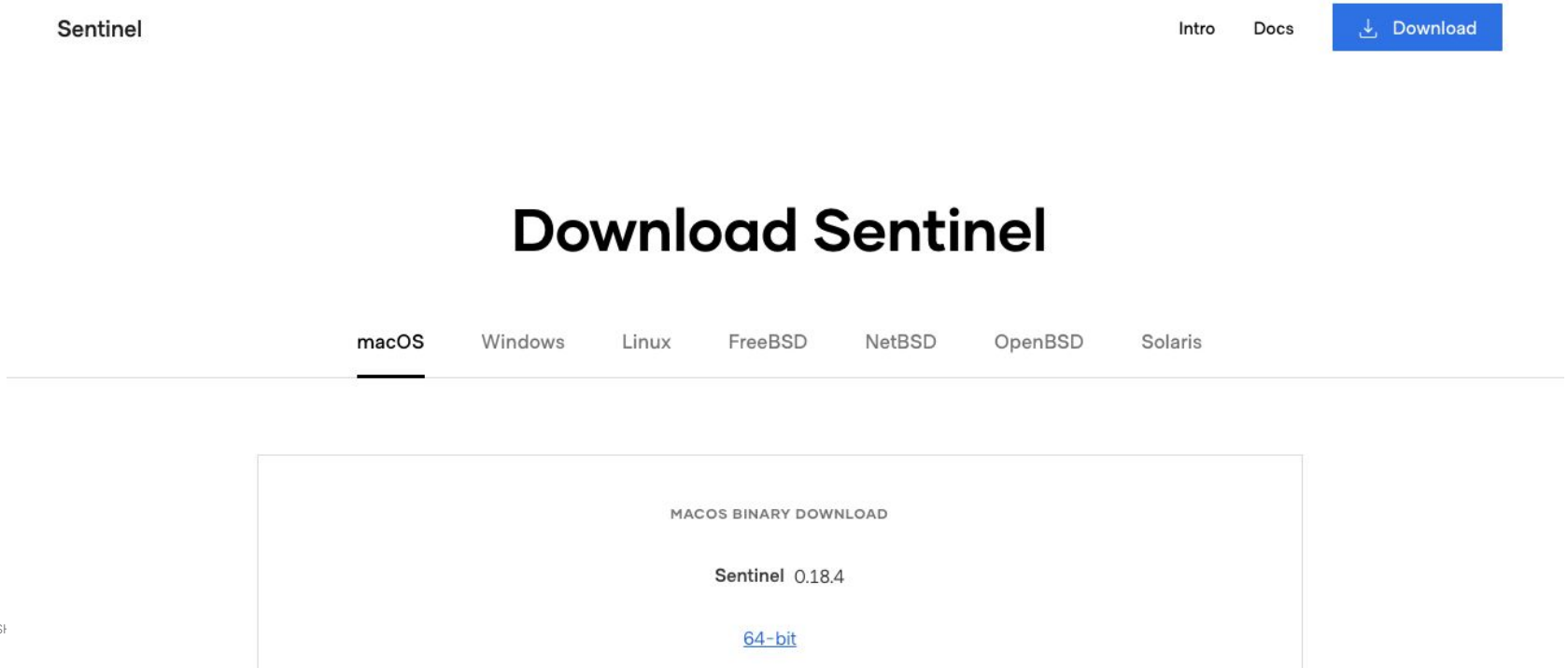
```
main = rule {
  identity.entity.name is "vincent" or

  identity.entity.id is
"fe2a5bfd-c483-9263-b0d4-f9d345c0ffee" or

  "sysops" in identity.groups.names or

  "c0ffee0a-5c07-4b97-81ec-0d423accb8e0" in
keys(identity.groups.by-id)

}
```

Policy evaluation workflow



Test policies with Sentinel CLI

[Download Sentinel]



Test cases 1/3

- Sentinel expects a **test/<policy_name>** folder with test case files in either HCL or JSON format
- Test case files contain data to test the policy

```
$ tree
├── cidr-check.sentinel
└── test
    └── cidr-check
        ├── fail.hcl
        └── success.hcl
```

Test cases 2/3

Specify the data to test the policy against

Optional: Expected boolean value of the rules

In absence of the ‘test’ block, all rules are expected to return true

```
global "my_global_variable" {
  value = <test_data>
},


test {
  rules = {
    <expected_result>
  }
}
```

Test cases 3/3

Use **mock** instead of global to inject static value directly into the policy's scope

For example, if the **time** library is used in the policy, use **mock** to mock **time.now**

```
mock "time" {
  data = {
    now = {
      weekday_name = "Monday"
      hour         = 14
    }
  }
}
```

```
import "sockaddr"

import "strings"

# Only evaluated for create, update, and delete operations against kv/ path

precond = rule {

    request.operation in ["create", "update", "delete"] and

    strings.has_prefix(request.path, "kv/")

}

# Requests must originate from our private IP range

cidrcheck = rule {

    sockaddr.is_contained(request.connection.remote_addr, "122.22.3.4/32")

}

# Check the precondition before executing the cidrcheck

main = rule when precond {

    cidrcheck

}
```

Passing test

The file test/cidr-check/success.hcl contains data for a passing test

```
global "request" {
  value = {
    connection = {
      remote_addr = "122.22.3.4"
    }
    operation = "create"
    path = "kv/orders"
  }
}
```

Failing test

The file test/cidr-check/fail.hcl contains data for a failing test

```
global "request" {
  value = {
    connection = {
      remote_addr = "122.22.3.10"
    }
    operation = "create"
    path = "kv/orders"
  }
}
test {
  rules = {
    main    = false
    precond = true
  }
}
```

Run tests

Use the **sentinel test** command to invoke the simulator and test policy

TIP: Use **-verbose** flag to output additional traces and logs for failed tests

```
$ sentinel test

PASS - cidr-check.sentinel
PASS - test/cidr-check/success.hcl
PASS - test/cidr-check/fail.hcl
```

Use CLI to deploy policy

Use vault CLI to write the policy

When successfully written, Vault begins immediately enforcing the policy at the hard mandatory level

```
$ vault write sys/policies/egp/cidr-check \
        policy=@cidr-check.sentinel \
        enforcement_level="hard-mandatory" \
        paths="kv/"

Success! Data written to: sys/policies/egp/cidr-check
```

02

Control Groups

Vault Control Groups

Allow for additional authorizations to be required for access to a path in Vault

When a control group is defined, the following occurs:

1. The requestor receives a wrapping token in return
2. The authorizers required by the control group policy must approve the request
3. Once all authorizations are satisfied, the requester can unwrap the secrets

Factors

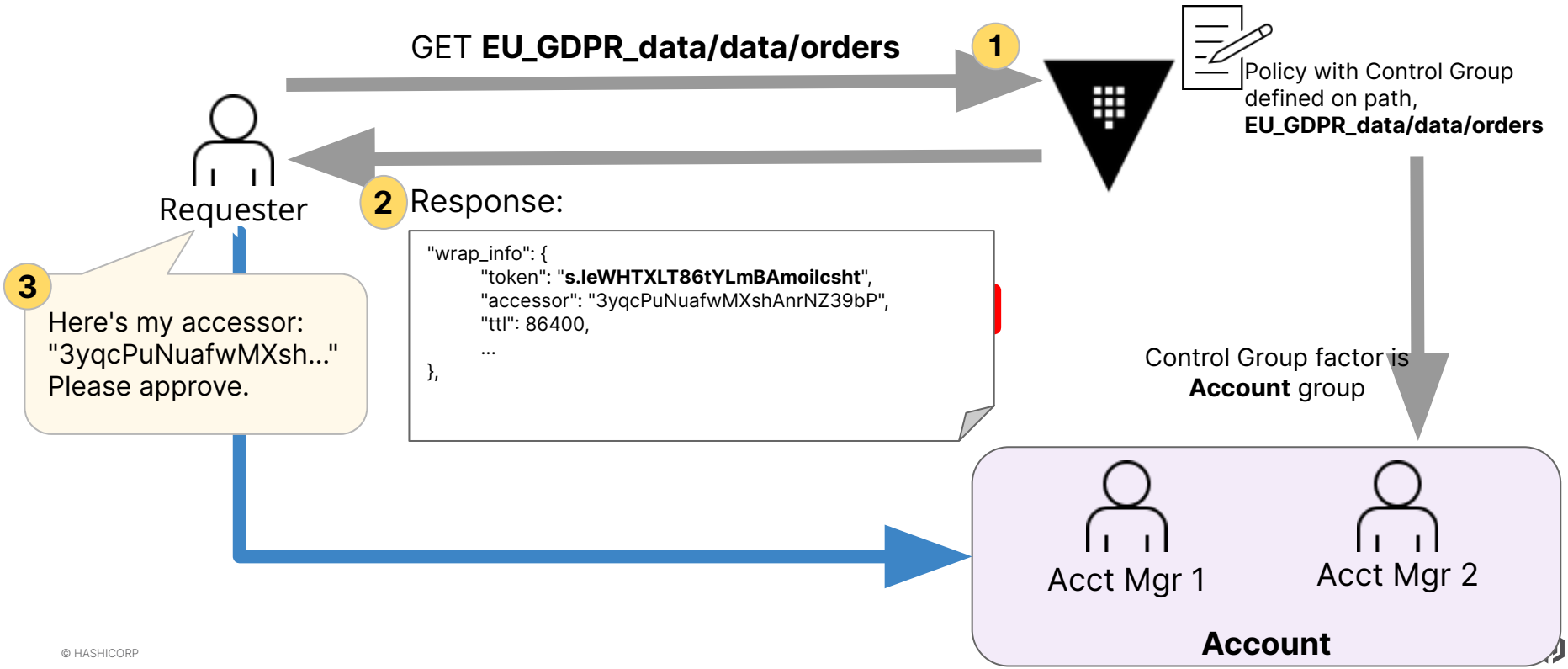
Control Group requirements on paths can be specified in:

- ACL policies
- Sentinel policies

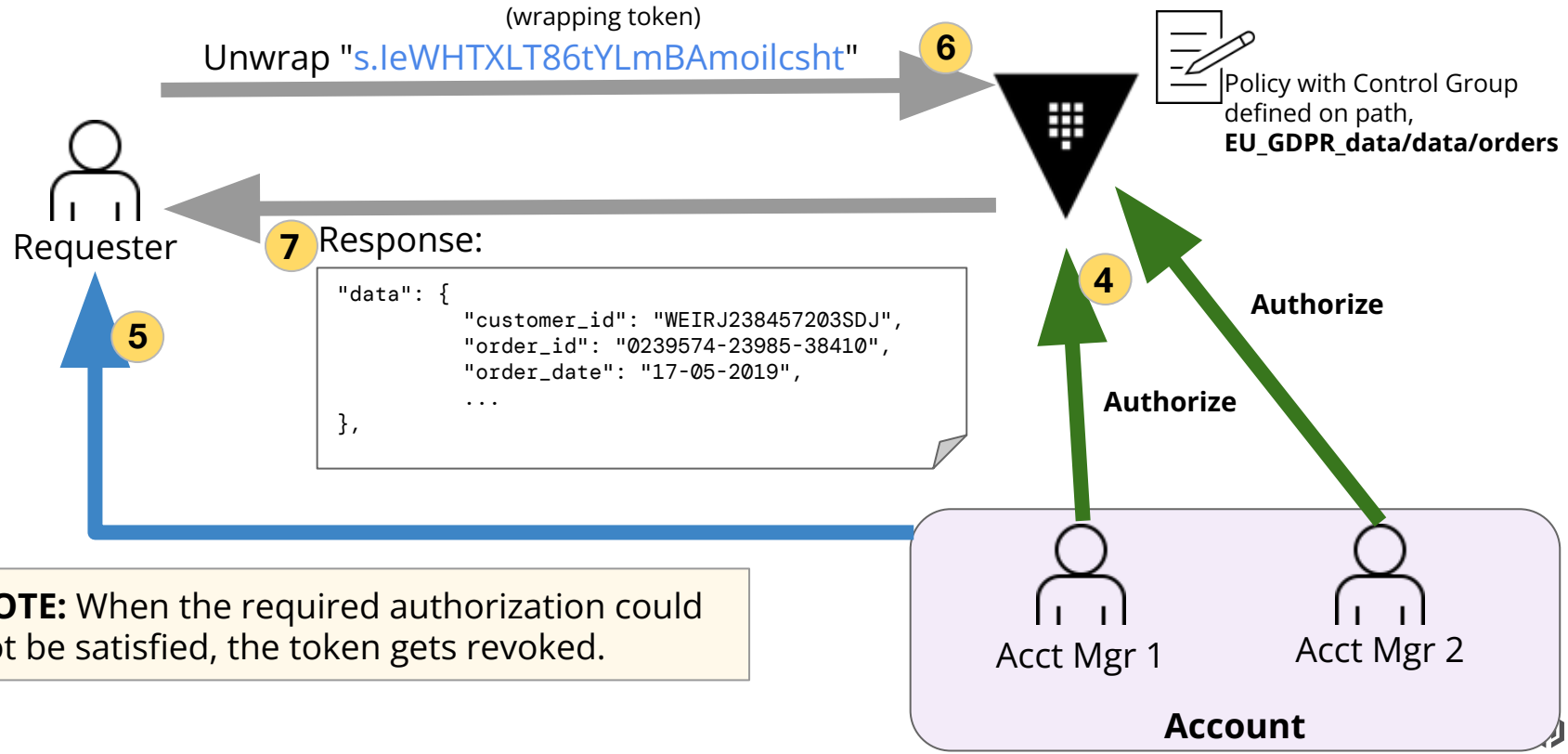
The single currently supported Control Groups factor is Identity Groups

- An authorizer must belong to a specific Identity group (factor)

Control Groups Workflow (1/2)



Control Groups Workflow (2/2)



NOTE: When the required authorization could not be satisfied, the token gets revoked.

Control Groups in ACL policies

Control Groups in Sentinel policies

```
import "controlgroup"

control_group = func() {
    numAuthzs = 0
    for controlgroup.authorizations as authz {
        if "account" in authz.groups.by_name {
            numAuthzs = numAuthzs + 1
        }
    }
    if numAuthzs >= 2 {
        return true
    }
    return false
}

main = rule {
    control_group()
}
```

Multiple factor Control Groups

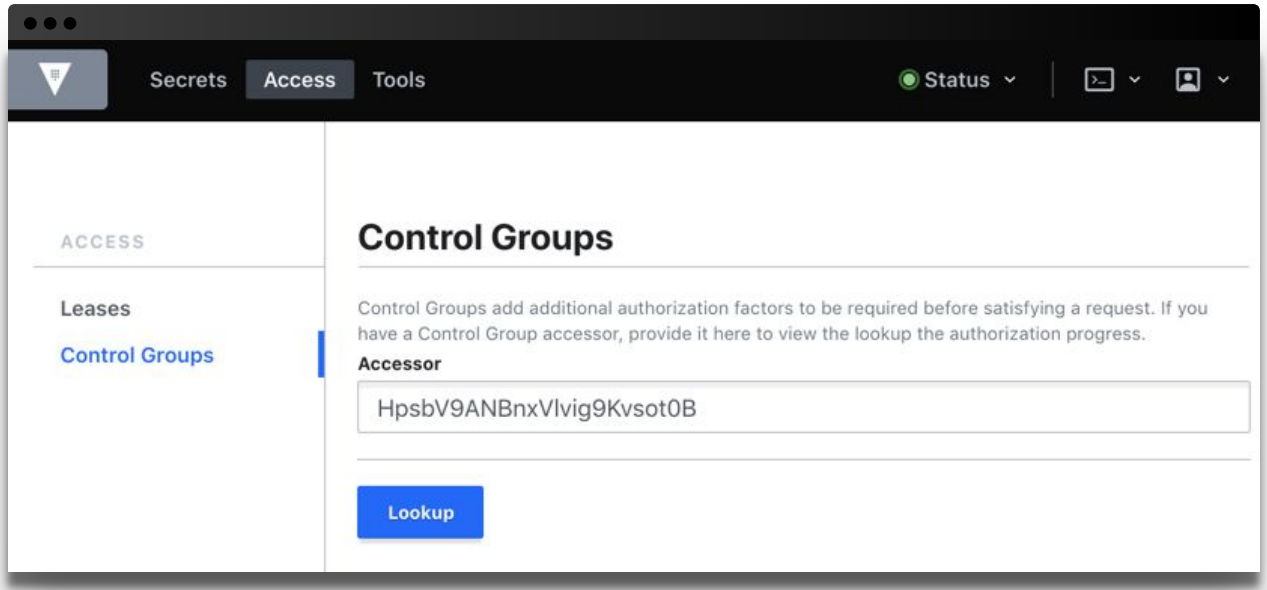
```
path "EU_GDPR_data/data/orders/*" {
    capabilities = [ "create",  "read", "update" ]

    control_group = {
        factor "acct_manager" {
              ttl = "4h"
               identity {
                   group_names = [ "account" ]
                    approvals = 2
               }
        }
       factor "security" {
               identity {
                   group_names = [ "eu-security" ]
                    approvals = 1
               }
        }
    }
}
```

Control Groups Workflow

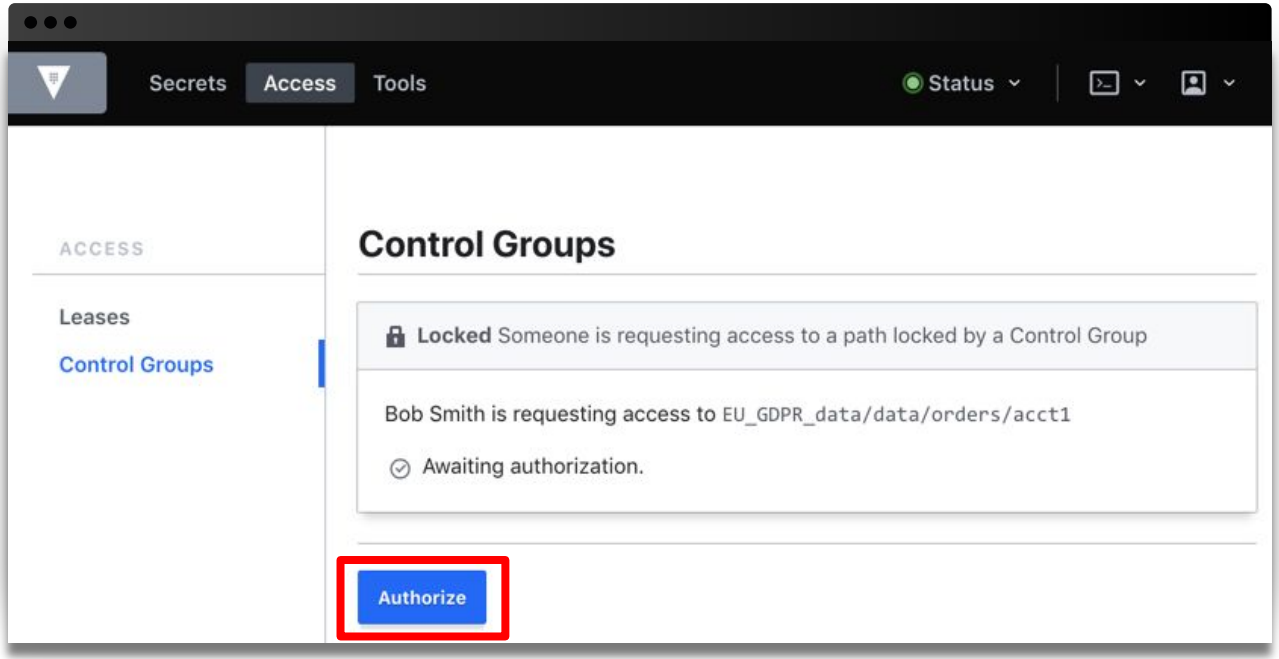
Requester actions

```
$ vault login -method=userpass username=bob
password=training

Key                    Value
—                      -----
Token                  s.5VvDmKAFZyxZ3tfBAhVntanm
…                      …
Token_policies         ["default"]
Identity_policies           ["read-gdpr-order"]

$ vault kv get EU_GDPR_data/orders/acct1

Key                    Value
---                    -----
wrapping_token:        s.JwkWZlsWIChFNaZSwDuMeF0A
wrapping_accessor:          HpsbV9ANBnxVlvig9Kvsot0B

wrapping_token_ttl:    24h
wrapping_token_creation_time:    2019-05-17 12:20:06 -0700
PDT
wrapping_token_creation_path:
EU_GDPR_data/data/orders/acct1
```

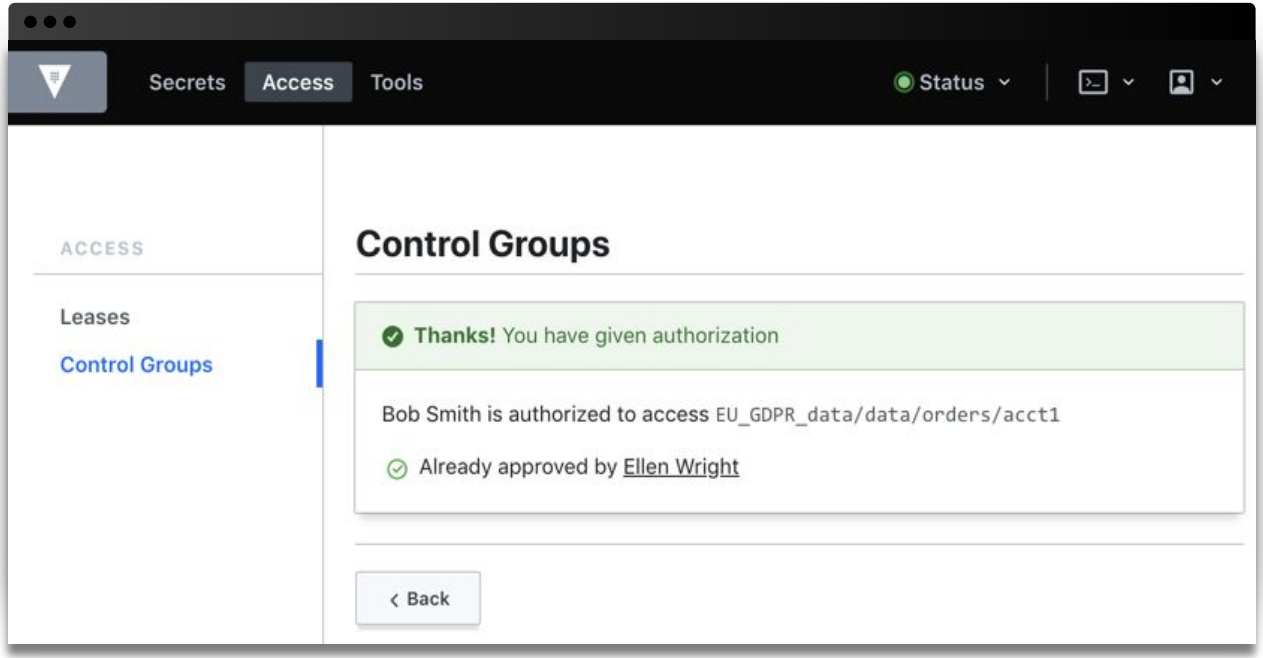
Authorizer actions



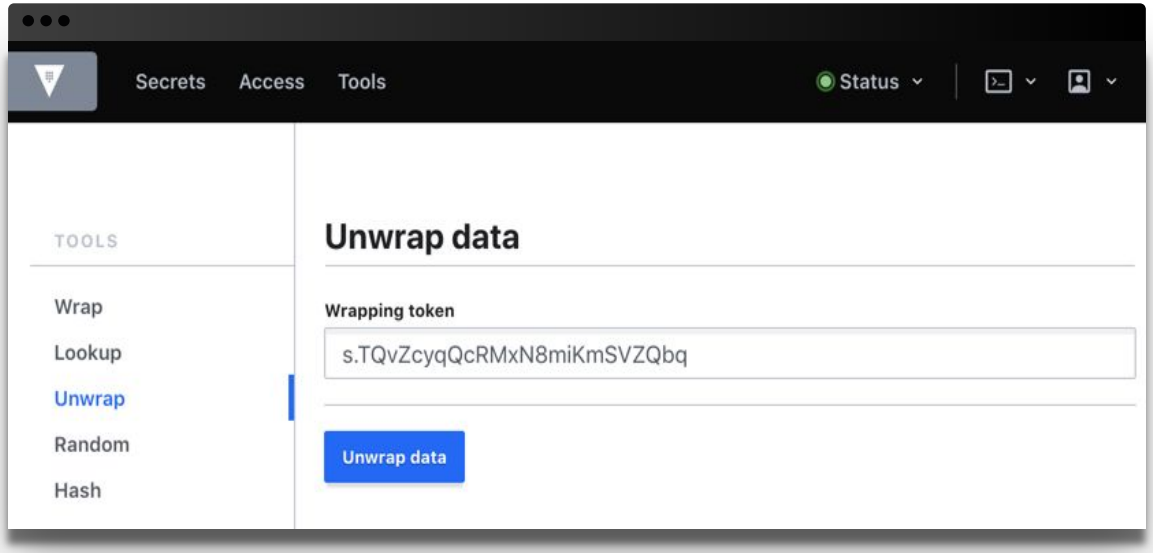
Authorizer actions



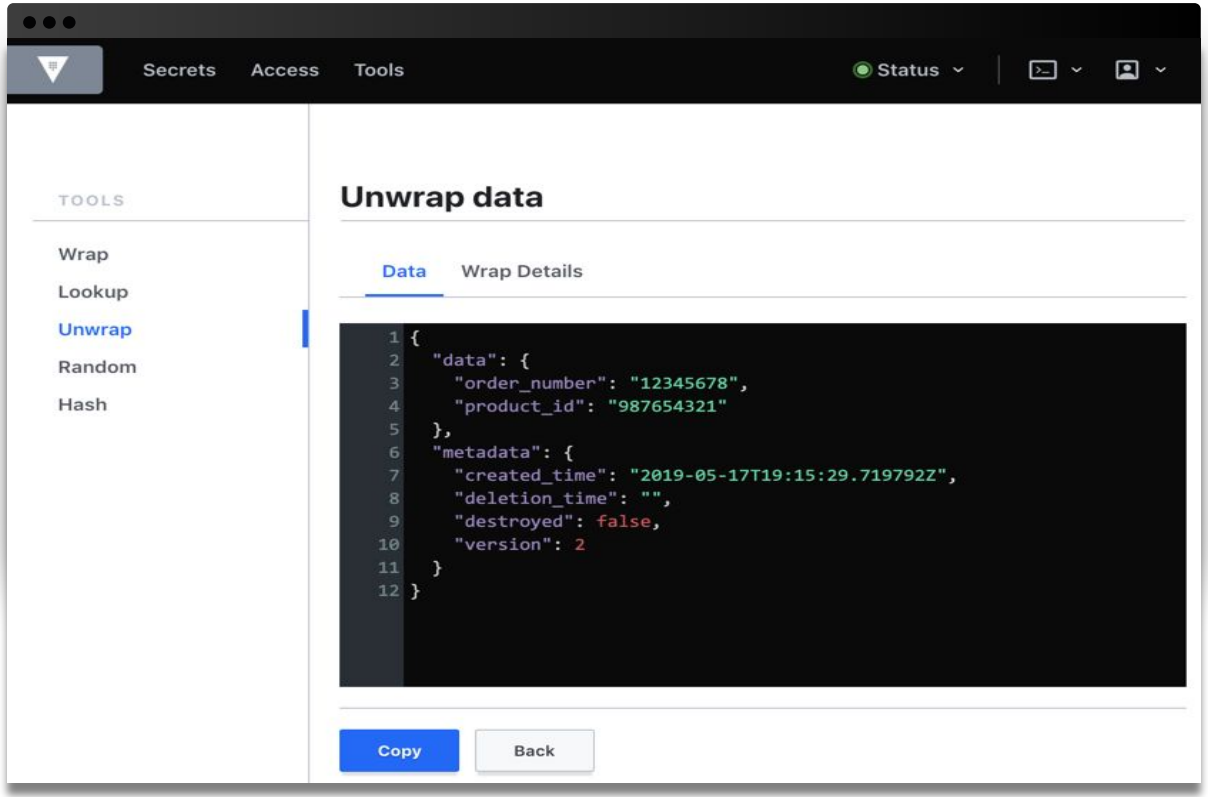
Authorizer actions



Unwrap the secret



Unwrap the secret



03

Quotas

Overview

Protect system stability and network, storage resource consumption from runaway application behavior and Distributed Denial of Service (DDoS)

- **Rate Limit Quotas**
  Limit maximum amount of requests per second (RPS) to a system or mount to protect network bandwidth

- **Lease Count Quotas**
  Cap number of leases generated in a system or mount to protect system stability and storage performance at scale

Configuring Resource Quotas

Creating Resource Quotas:

To set rate limit or lease quotas, set /sys/quotas/**<type>** with possible types being:

- rate-limit: Rate limit quota
- lease-count: Maximum lease count (Enterprise only)

Configuring Resource Quotas:

- Set/list specific types of quotas from **/sys/quotas/<type>/<name>**
- Path parameter can be set to a mount, mount in the namespace, or omitted for a systemwide quota
- Different quotas have different parameters; see the documentation for more details

Logging

- If a request is rejected due to a ‘Lease Count Quota’ violation, Vault will record this in the audit log
- Violations of ‘Rate Limit Quotas’ are not logged to the audit log
- It is possible to enable traceability of ‘Rate Limit Quotas’ via the HTTP API endpoint: sys/quotas/config, changing the parameter: ‘enable_rate_limit_audit_logging’ to **true** (by default it will be false)
- Enabling this can potentially impact performance if the request volume is large

Rate Limit Quotas Example

Protect Vault from potential DDoS attack

```
# Create global quota rule and set rate at desired
requests per second
$ vault write sys/quotas/rate-limit/global-rate
rate=500

# Set rate limit on specific paths
$ vault write sys/quotas/rate-limit/db-creds rate=30
      path="database"

# Set rate limit on specific namespace and path
$ vault write sys/quotas/rate-limit/orders \

     path="us-west/kv-v2" \

     rate=16.67 \

     burst=100
```

Lease Count Quotas Example

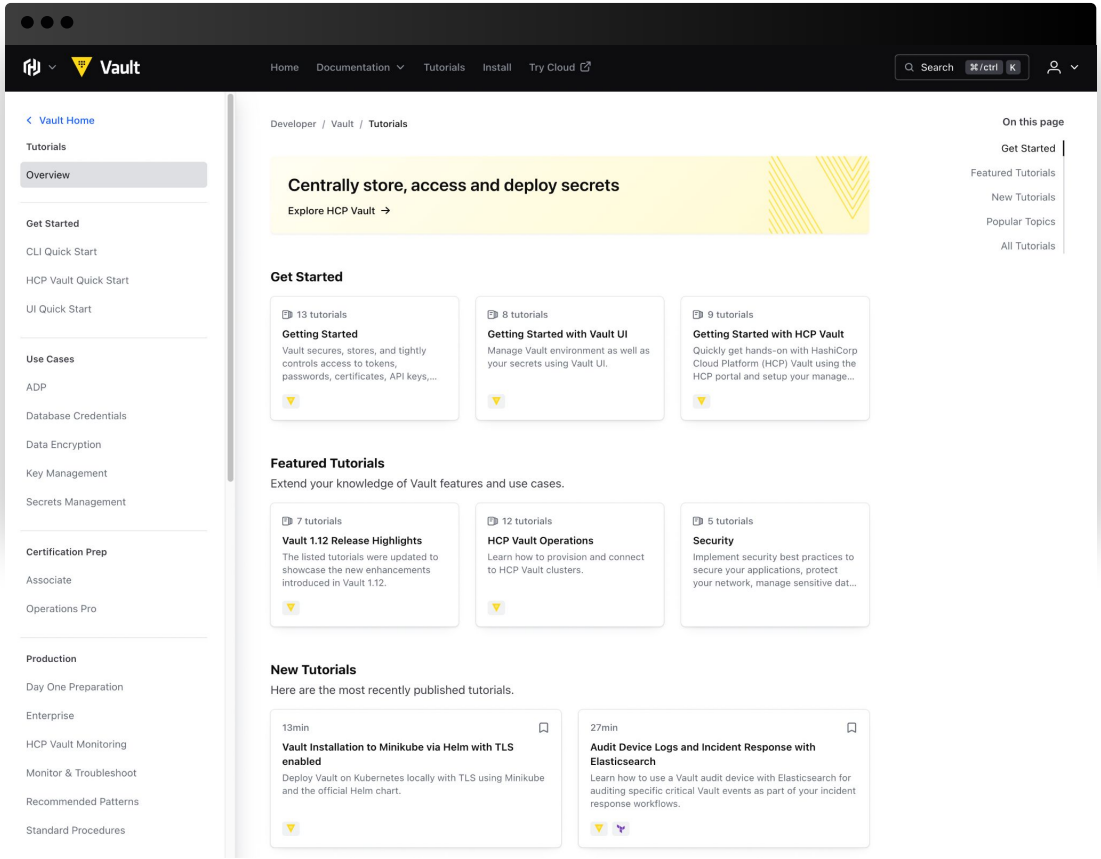
Prevent the storage backend from becoming the point of failure

```
$ vault write
sys/quotas/lease-count/global-count-limit \
     max_leases=500


$ vault write sys/quotas/lease-count/db-creds \
     max_leases=100
     path="us-west/postgres"
```

Next Steps

Tutorials

Step-by-step guides to accelerate deployment of Vault



https://developer.hashicorp.com/vault/tutorials

Resources

- Sentinel Documentation
- Sentinel Properties
- Sentinel Policy Examples
- Standard Sentinel Imports
- List of Vault data available for Sentinel policies
- Sentinel Policy Tutorial
- Control Groups Tutorial
- Protecting Vault with Resource Quotas

Need Additional Help?

Customer Success

Contact our Customer Success Management team with any questions. We will help coordinate the right resources for you to get your questions answered

customer.success@hashicorp.com

Technical Support

Something not working quite right? Engage with HashiCorp Technical Support by opening a ticket for your issue at

support.hashicorp.com

Discuss

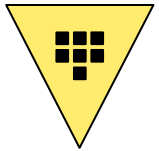
Engage with the HashiCorp Cloud community including HashiCorp Architects and Engineers

discuss.hashicorp.com

Action Items

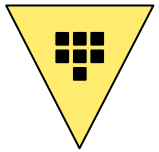
- Share to customer.success@hashicorp.com
  - Authorized technical contacts for support
  - Stakeholders contact information (name and email addresses)
- Review where Sentinel and/or Control Groups are appropriate for your environment(s)
- Implement quotas for active clusters

Upcoming Webinars



Program Closing

Asynchronous content that will be delivered to your Inbox



Please complete the Zoom survey when prompted at the end of today's webinar

Q&A

Thank you

customer.success@hashicorp.com

www.hashicorp.com/customer-success