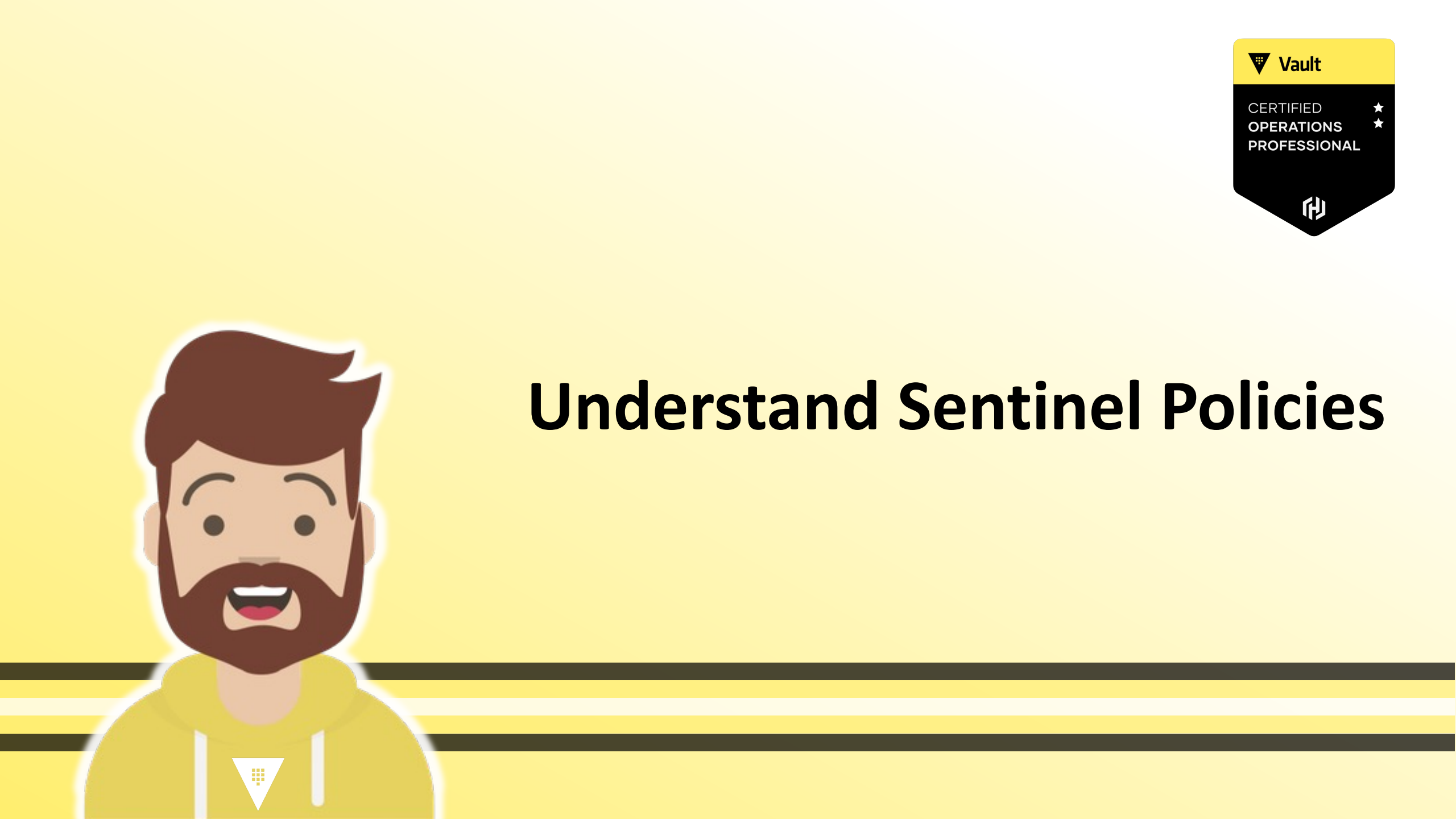


Understand Sentinel Policies

What is Sentinel?

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.

Policy as Code

Treat policy like an application — version control, pull review, and automate tests. Use programming constructs to determine policy decisions beyond the limited constraints of typical ACL systems.

Fine Grained, Conditioned-Based

Treat policy like an application — version control, pull review, and automate tests. Use programming constructs to determine policy decisions beyond the limited constraints of typical ACL systems.

Embedded

Sentinel is embedded to enable policy enforcement in the data path to actively reject violating behavior instead of passively detecting.

Enforcement Levels

Advisory, soft-mandatory, and hard-mandatory levels allow policy writers to warn on or reject offending behavior.

External Information

Sentinel can permit or deny actions based upon external information available to the token, such as time, IP address, requested path, etc.

Multi-Cloud Compatible

Ensure infrastructure changes are within business and regulatory policy on every infrastructure provider.

Multi-Platform

Sentinel is NOT just a Vault feature.

It is available in the Enterprise versions of other HashiCorp Products.

HashiCorp Terraform Enterprise

HashiCorp Nomad Enterprise

HashiCorp Vault Enterprise

HashiCorp Consul Enterprise

Types of Sentinel Policies

Role Governing Policies (RGPs)

- Sentinel policies that are tied to **tokens**, **identity entities**, or **identity groups**
- Access to rich set of controls across various aspects of Vault

Endpoint Governing Policies (EGPs)

- Sentinel policies that are tied to **paths** instead of tokens
- Access to as much request information as possible
  - Can take an effect even on unauthenticated paths (e.g., login paths)

Anatomy of a Sentinel Policy

- **Import** – access to reusable libraries to import information or use features
- **Main** – (required) the main rule to be evaluated
- **Rule** – describes a set of conditions resulting in either true or false
- **Variables** – optional, dynamically typed variable

```
import "<library>"
<variable> = <value>
<name> = rule { <condition_to_evaluate> }
main = rule {
      <condition_to_evaluate>
}
```

Imports

Example of **Imports** that can be used with Sentinel:

- **`base64`** – encode & decode Base64 values
- **`decimal`** – provides functions for operating on numbers as decimals
- **`http`** – enables the use of HTTP-accessible data outside of the runtime in Sentinel rules
- **`json`** – parse and access a JSON document
- **`runtime`** – contains various information about Sentinel runtime
- **`sockaddr`** – enables working with IP addresses
- **`strings`** – enables common string operations
- **`time`** – provides access to execution time and time functions
- **`types`** – ability to parse an object's type
- **`units`** – provides access to quick calculations for various byte units
- **`version`** – used to parse versions and version constraints

These allow fine-grained controls over your Vault environment

https://docs.hashicorp.com/sentinel/imports

Sentinel Policy Example - RGP

Only allow a specific entity or groups

```
main = rule {
    identity.entity.name is "jeff" or
      identity.entity.id is "fe2a5bfd-c483-9263-b0d4-f9d345efdf9f" or
      "sysops" in identity.groups.names or
      "14c0940a-5c07-4b97-81ec-0d423accb8e0" in keys(identity.groups.by-id)
}
```

If the user "Jeff" is deleted and recreated, the match will fail because we're also enforcing the entity ID

Sentinel Policy Example - EGP

Disallow all previously-generated tokens based on date:

- You could apply this EGP to the "*" endpoint

```
import "time"

main = rule when not request.unauthenticated {
    time.load(token.creation_time).unix >
        time.load("2022-12-25T00:00:01Z").unix
}
```

Could be used as a "break-glass" scenario where previous tokens were compromised

Sentinel Policy Example - EGP


```
import "sockaddr"
import "mfa"
import "strings"

# We expect logins to come only from a specific private IP range
cidrcheck = rule {
    sockaddr.is_contained(request.connection.remote_addr, "10.0.23.0/16")
}

# Require Ping MFA validation to succeed
ping_valid = rule {
    mfa.methods.ping.valid
}

main = rule when request.path is "auth/ldap/login" {
    ping_valid and cidrcheck
}
```

Sets the scope of policy

Must also pass both rules

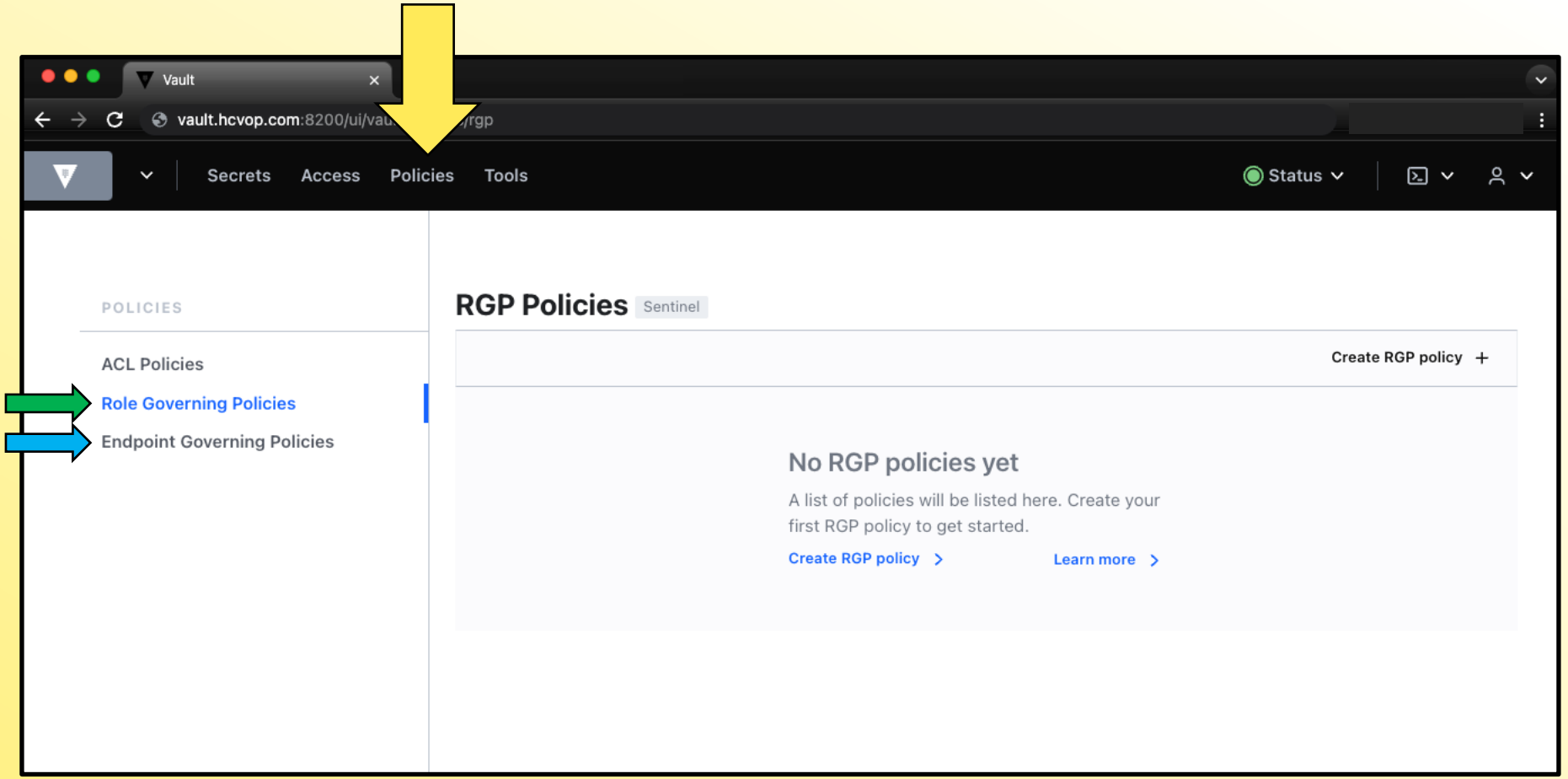
Enforcement Levels

Sentinel offers three different enforcement levels that can be set per Sentinel policy:

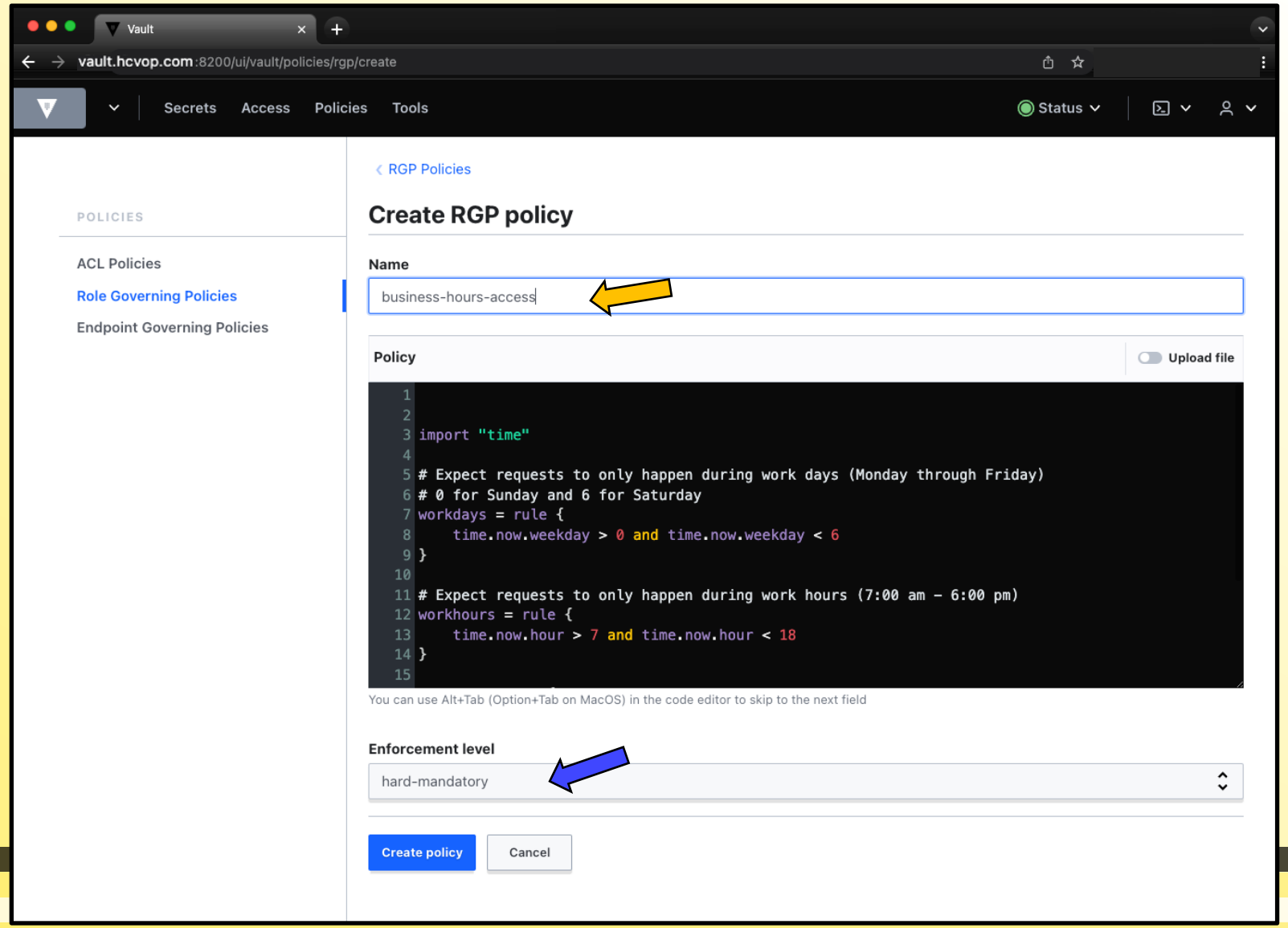
Enforcement Level	Description
Advisory	The policy is allowed to fail
Soft Mandatory	The policy must pass unless an override is specified
Hard Mandatory	The policy muss pass no matter what

To override a Sentinel policy (soft mandatory), use the –policy-override flag when executing the Vault command

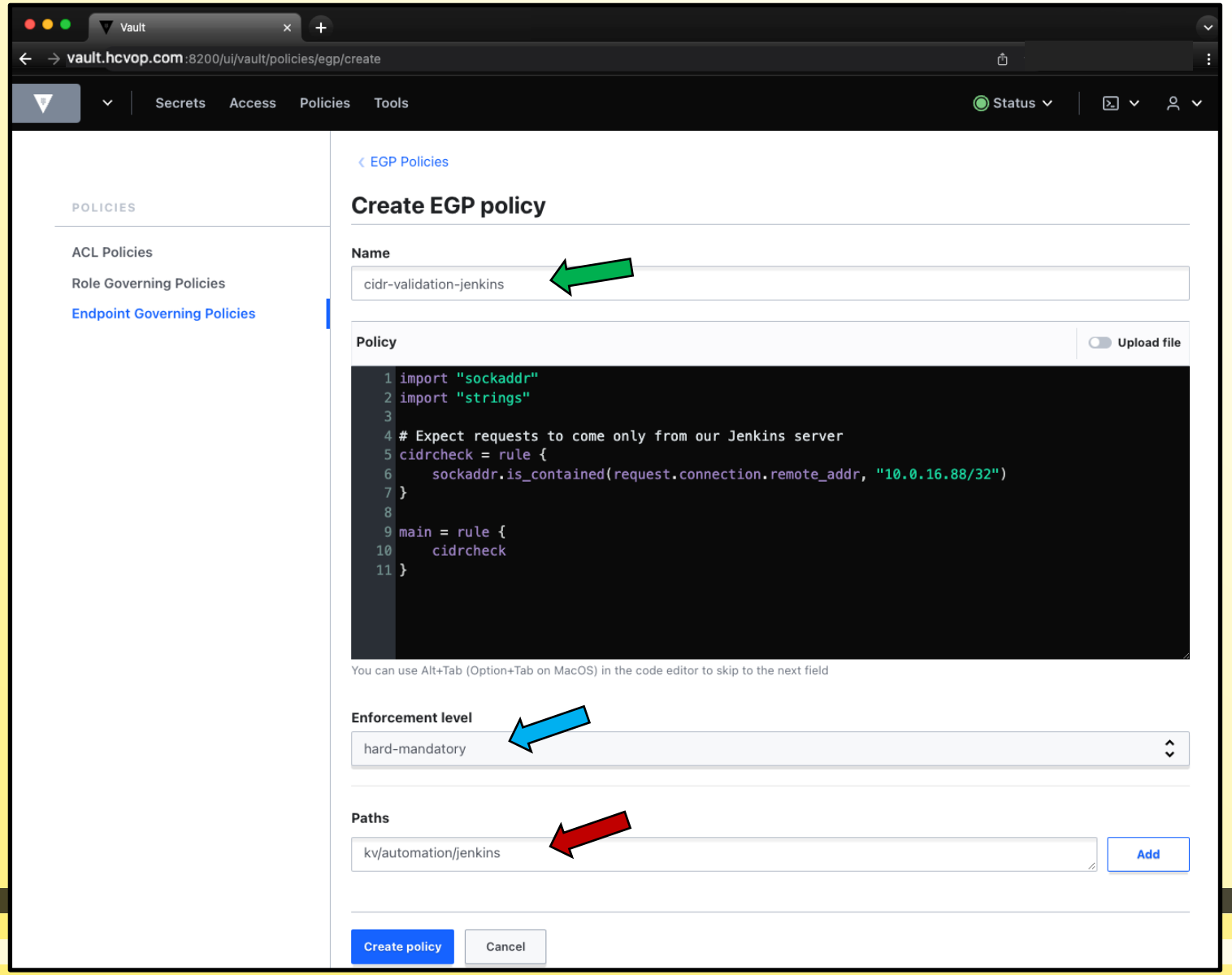
Deploy Sentinel Policies via UI



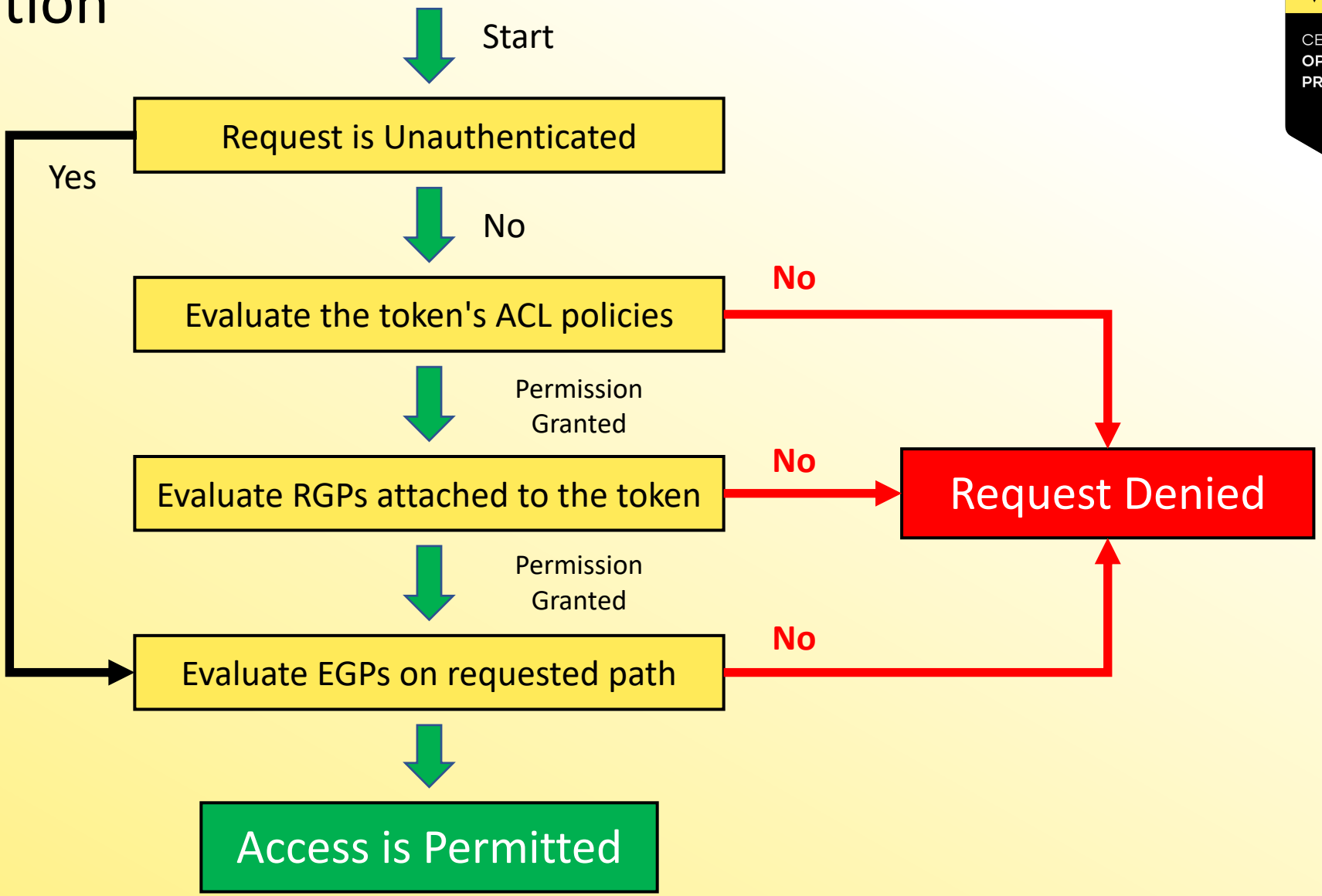
Deploy RGP Sentinel Policy via UI



Deploy EGP Sentinel Policy via UI



Policy Evaluation



END OF SECTION