
# Securely Configure Auto-Auth and Token Sink

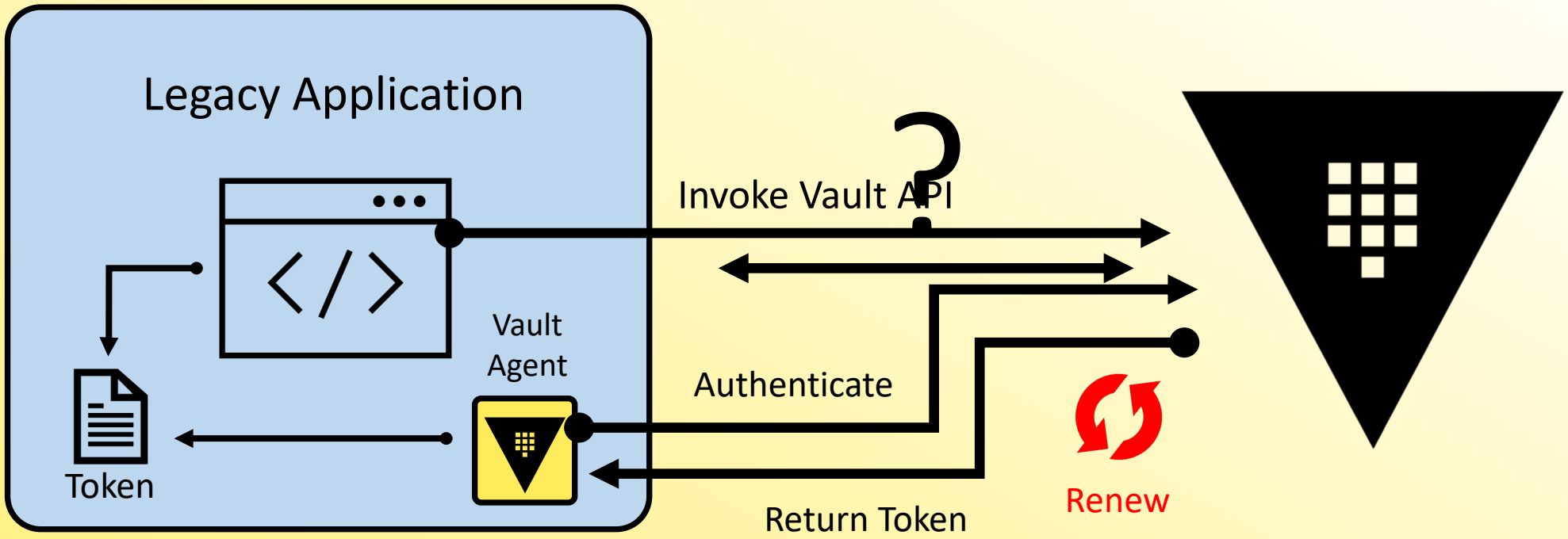
# What is the Vault Agent?


The Vault Agent is a client daemon that runs alongside an application to enable legacy applications to interact and consume secrets

Vault Agent provides several different features:

- Automatic authentication including renewal
- Secure delivery/storage of tokens (response wrapping)
- Local caching of secrets
- Templating

# Legacy Applications – Auto-Auth



# Vault Agent – Auto-Auth


- The Vault Agent uses a pre-defined auth method to authenticate to Vault and obtain a token
- The token is stored in the "sink", which is essentially just a flat file on a local file system that contains the Vault token
- The application can read this token and invoke the Vault API directly
- This strategy allows the Vault Agent to manage the token and guarantee a valid token is always available to the application

# Vault Agent - Auto Auth


Vault Agent supports many types of auth methods to authenticate and obtain a token

Auth methods are generally the methods you'd associate with "machine-oriented" auth methods

- **`AliCloud`**
- **`AppRole`**
- **`AWS`**
- **`Azure`**
- **`Certificate`**
- **`CloudFoundary`**
- **`GCP`**
- **`JWT`**
- **`Kerberos`**
- **`Kubernetes`**

# Vault Agent Configuration File

```
auto_auth {
  method "approle" {
      mount_path = "auth/approle"
      config = {
          role_id_file_path = "<path-to-file>"
          secret_id_file_path = "<path-to-file>"
      }
  }

  sink "file" {
      config = {
          path = "/etc/vault.d/token.txt"
      }
  }
}

vault {
  address = "http://<cluster_IP>:8200"
}
```

**Auto-Auth Configuration (AppRole)**

**Sink Configuration**

**Vault**

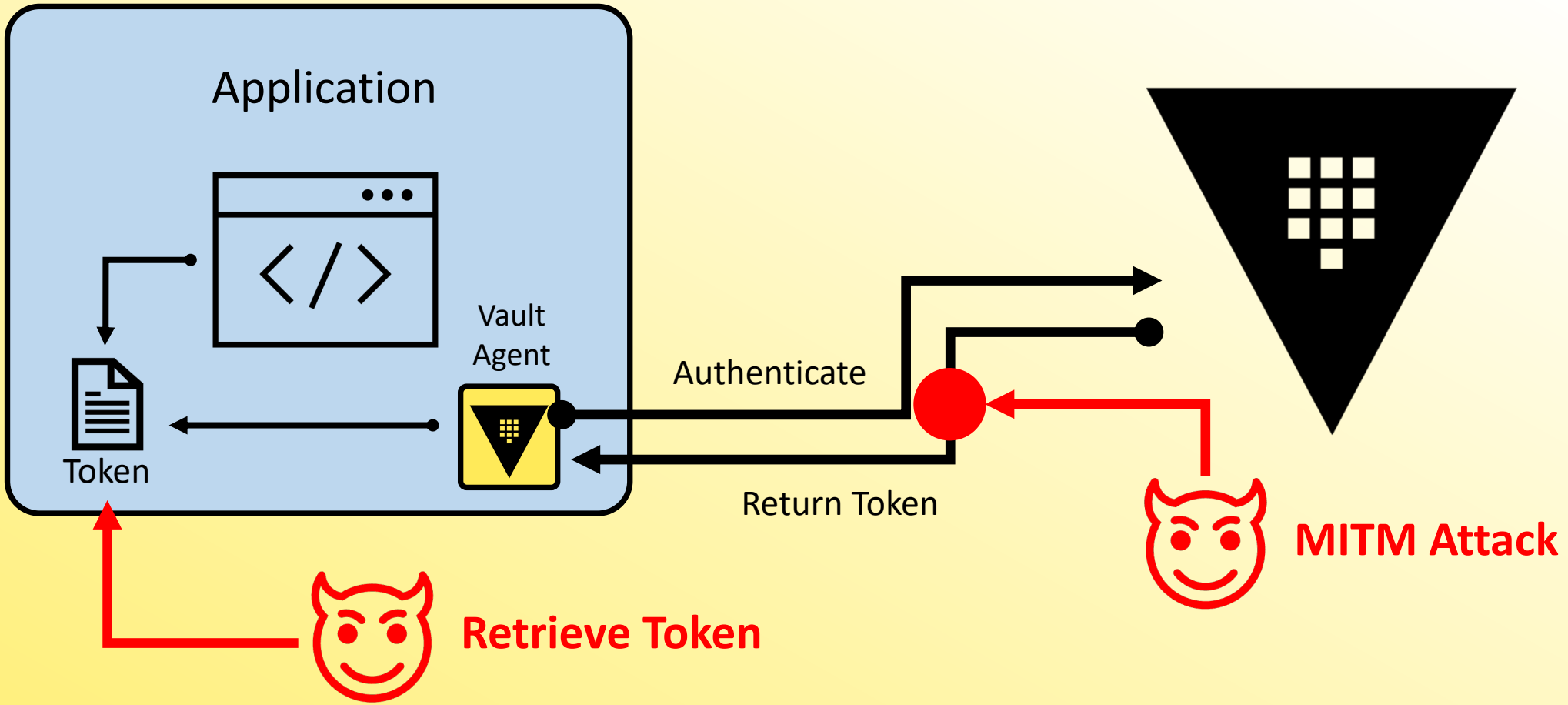
# Vault Agent - Sink

As of today, `file` is the only supported method of storing the auto-auth token

Common configuration parameters include:

- `type` – what type of sink to use (again, only `file` is available)
- `path` – location of the file
- `mode` – change the file permissions of the sink file (default is 640)
- `wrap_ttl` = retrieve the token using response wrapping

# Auth-Auth Security Concerns



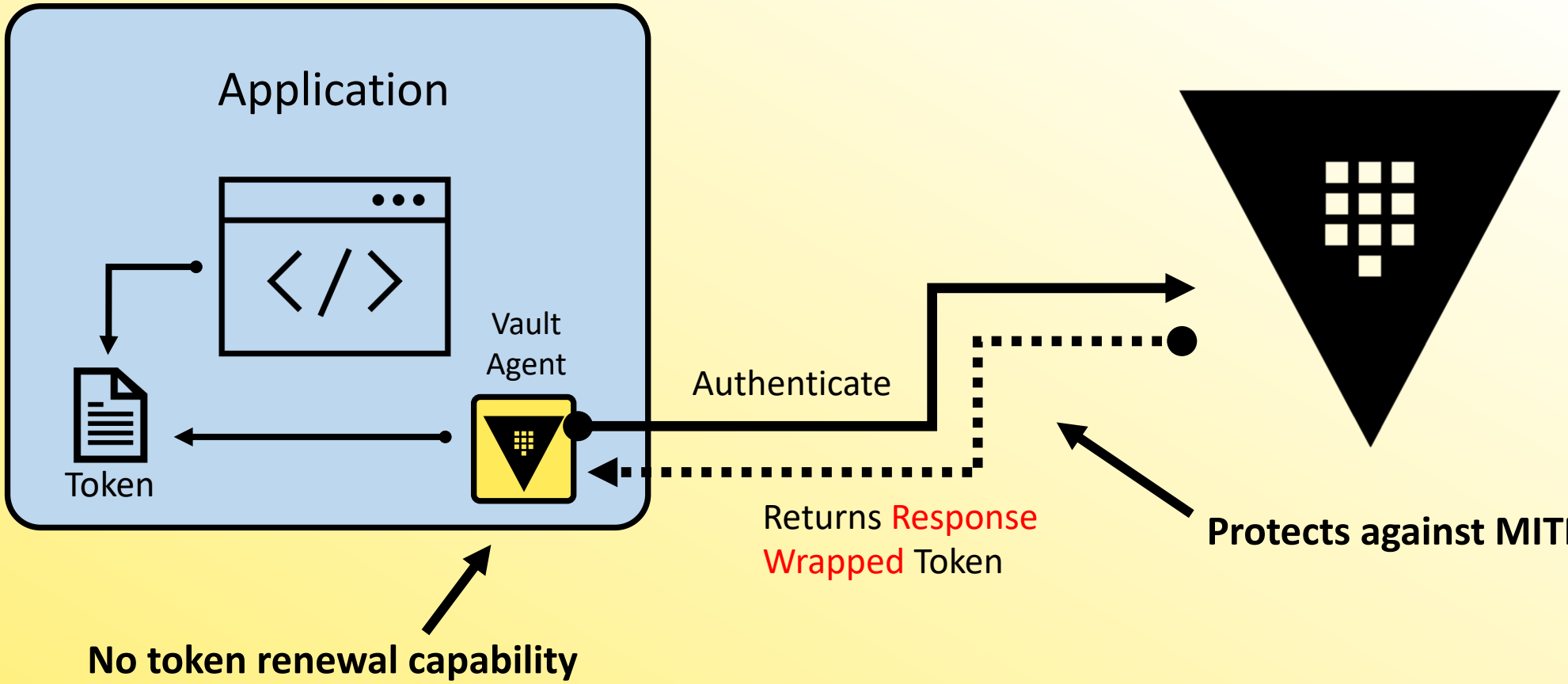
# Protecting the Token using Response Wrapping


To help secure tokens when using Auth-Auth, you can have Vault response wrap the token when the Vault Agent authenticates

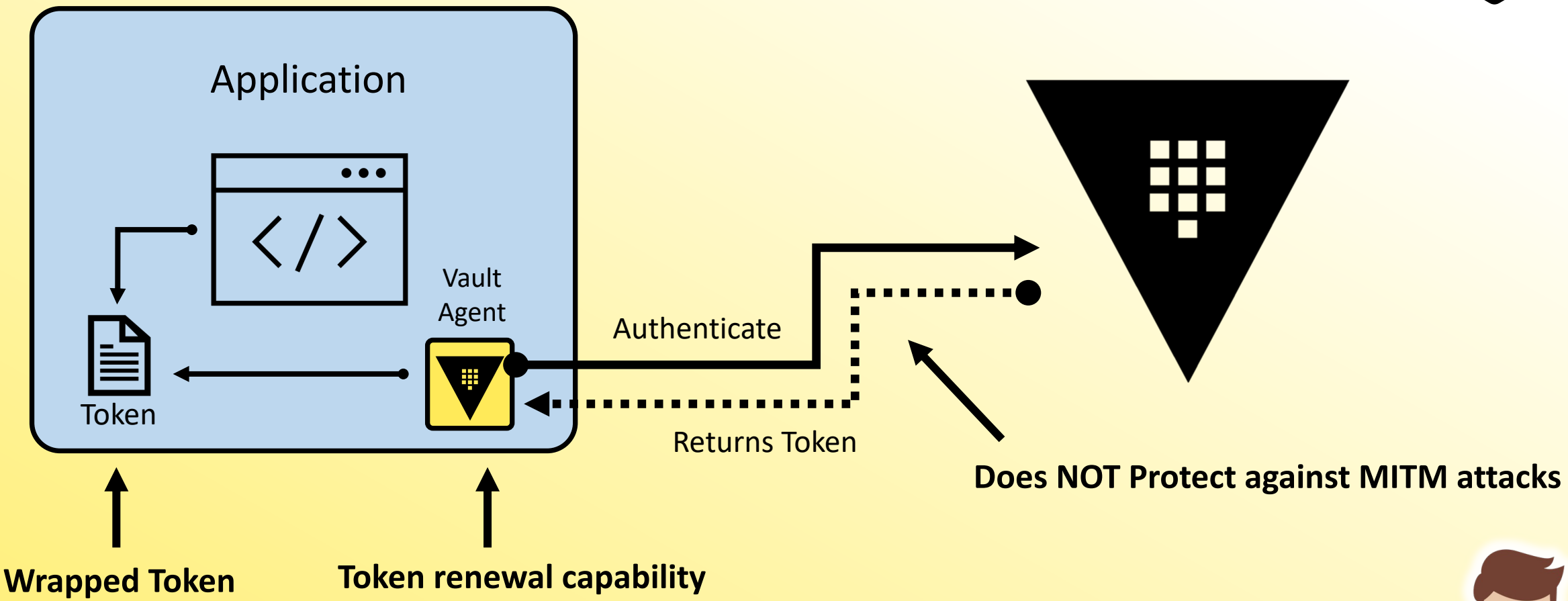
- Response wrapped by the auth method
- Response wrapped by the token sink

The placement of the **`wrap_ttl`** in the Vault Agent configuration file determines where the response wrapping happens.

# Response Wrapping at the **Auth Method**



# Response Wrapping at the **Sink**



# Response-Wrapping the Token - Comparison


## Response Wrapped by the Auth Method

**Pros:**

- Prevents man-in-the-middle attacks (MITM)
- More secure

**Cons:**

- Vault agent cannot renew the token

## Response Wrapped by the Sink

**Pros:**

- Allows the Vault agent to renew the token and re-authenticate when the token expires

**Cons:**

- The token gets wrapped ***after*** it is retrieved from Vault. Therefore, it is vulnerable to MITM attack

# Response-Wrapping the Token - Comparison

## Response Wrapped by the Auth Method

```
pid_file = "/home/vault/pidfile"

auto_auth {
   method "kubernetes" {
     wrap_ttl = "5m"
     mount_path = "auth/kubernetes"
     config = {
       role = "example"
     }
   }
...

vault {
  address = "http://<cluster_IP>:8200"
}
```

## Response Wrapped by the Sink

```
pid_file = "/home/vault/pidfile"

auto_auth {
  method "kubernetes" {
    mount_path = "auth/kubernetes"
     config = {
       role = "example"
     }
  }
   sink "file" {
      wrap_ttl = "5m"
      config = {
        path = "/etc/vault/token"
      }
   }
}
```

# END OF SECTION