Module 1: Vault Tokens

1.1 Overview of Tokens

Tokens are the core method for authentication within the Vault.

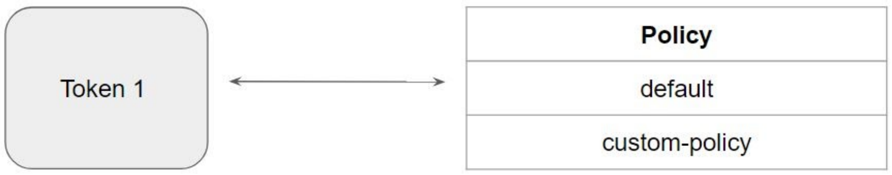
With **vault server -dev**, we have been using the root token. This is the first method of authentication for Vault. It is also the only auth method that cannot be disabled.

```
Key                     Value
---                     -----
token                   s.wzdOjDg6gUMRkmo9uF61XsqZ
token_accessor          sKOVJcPb7S9wQCDHeJeB2dXY
token_duration          768h
token_renewable         true
token_policies          ["default" "first-policy"]
identity_policies       []
policies                ["default" "first-policy"]
token_meta_username     admin
```

1.2 Mapping of Tokens to Policies

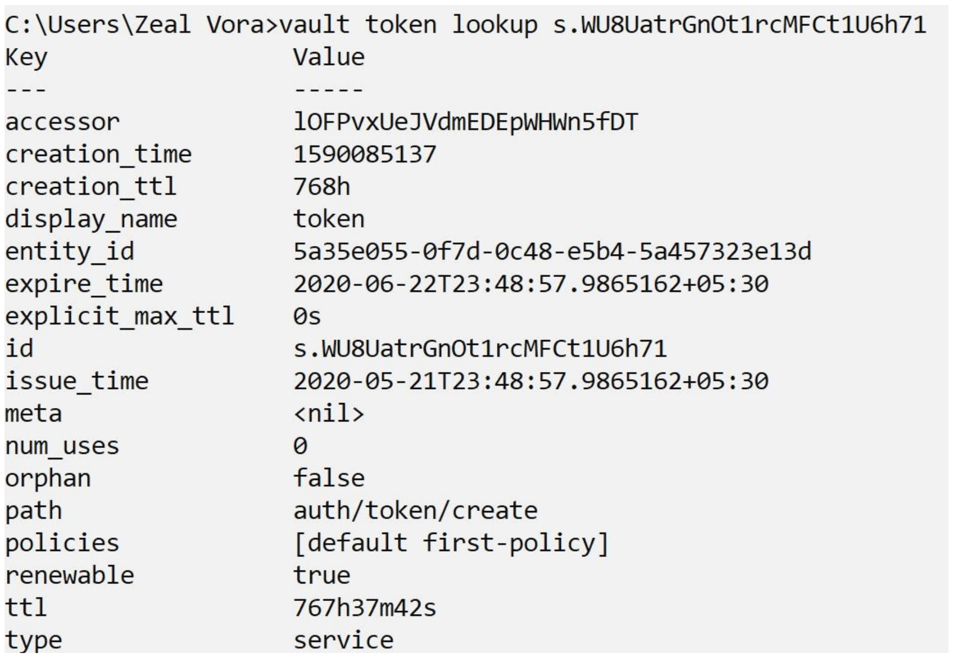
Within Vault, tokens map to information.

The most important information mapped to a token is a set of one or more attached policies.



1.3 Lookup Token Information

You can explore the details of a given token with the help of the **vault token lookup** command.



```
C:\Users\Zeal Vora>vault token lookup s.WU8UatrGnOt1rcMFCt1U6h71
Key                  Value
---                  -----
accessor             lOFPvxUeJVdmEDEpWHWn5fDT
creation_time        1590085137
creation_ttl         768h
display_name         token
entity_id            5a35e055-0f7d-0c48-e5b4-5a457323e13d
expire_time          2020-06-22T23:48:57.9865162+05:30
explicit_max_ttl     0s
id                   s.WU8UatrGnOt1rcMFCt1U6h71
issue_time           2020-05-21T23:48:57.9865162+05:30
meta                 <nil>
num_uses             0
orphan               false
path                 auth/token/create
policies             [default first-policy]
renewable            true
ttl                  767h37m42s
type                 service
```

1.4 Important Points to Remember

Every non-root token has a time-to-live (TTL) associated with it.

After the current TTL is up, the token will no longer function -- it, and its associated leases, are revoked.

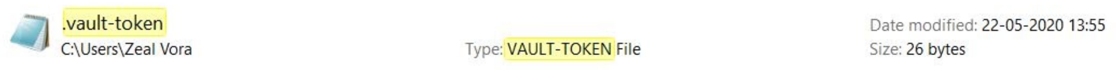
If the token is renewable, Vault can be asked to extend the token validity period using vault token renew

Root tokens can be non-expiring (those with a TTL of zero)

Module 2: Token Helper

By default, the Vault CLI uses a "token helper" to cache the token after authentication.

The default token helper stores the token in ~/.vault-token. You can delete this file at any time to "log out" of Vault.



.vault-token
C:\Users\Zeal Vora

Type: VAULT-TOKEN File

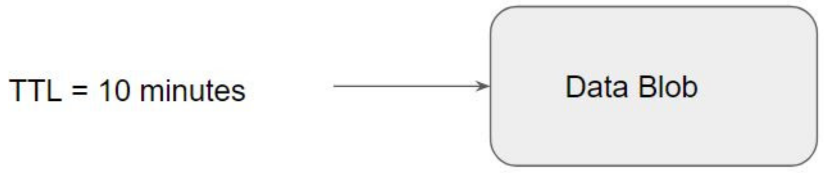
Date modified: 22-05-2020 13:55
Size: 26 bytes

Module 3: Token Time-to-Live

3.1 Overview of Time to Live

Time-To-Live (TTL) limits the lifetime of the data.

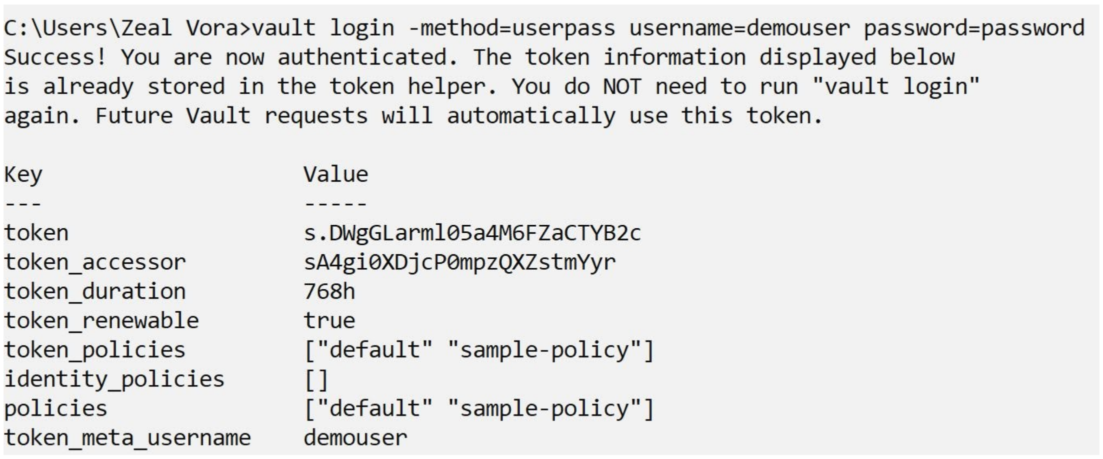
Used extensively in DNS.



3.2 TTL For Tokens

Every non-root token has a time-to-live (TTL) associated with it.

After the current TTL is up, the token will no longer function, and its associated leases, are revoked.

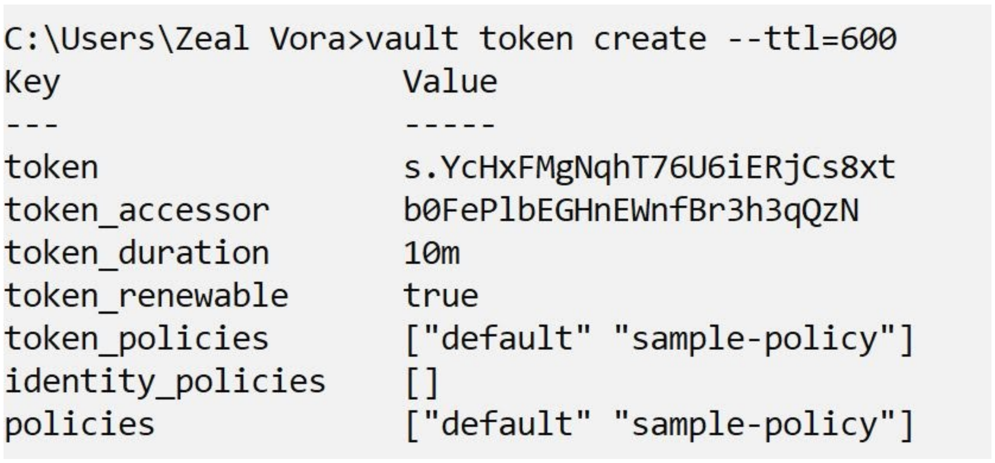


```
C:\Users\Zeal Vora>vault login -method=userpass username=demouser password=password
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                    Value
---                    -----
token                  s.DWgGLarml05a4M6FZaCTYB2c
token_accessor         sA4gi0XDjcP0mpzQXZstmYyr
token_duration         768h
token_renewable        true
token_policies         ["default" "sample-policy"]
identity_policies      []
policies               ["default" "sample-policy"]
token_meta_username    demouser
```

3.3 Create Token with Explicit TTL

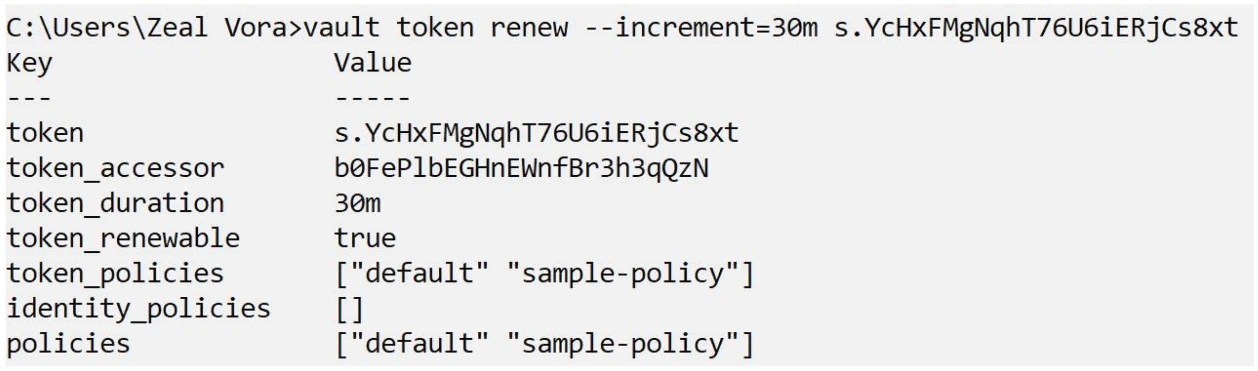
You can create a token that has a user-defined TTL



```
C:\Users\Zeal Vora>vault token create --ttl=600
Key                    Value
---                    -----
token                  s.YcHxFMgNqhT76U6iERjCs8xt
token_accessor         b0FePlbEGHnEWnfBr3h3qQzN
token_duration         10m
token_renewable        true
token_policies         ["default" "sample-policy"]
identity_policies      []
policies               ["default" "sample-policy"]
```

3.4 Renewing a Token

vault token renew command can be used to extend the validity of the renewable tokens.

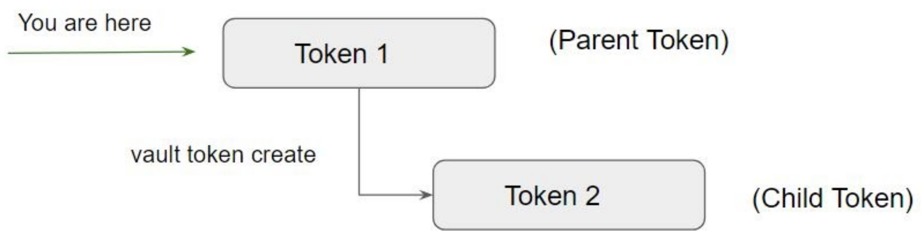


```
C:\Users\Zeal Vora>vault token renew --increment=30m s.YcHxFMgNqhT76U6iERjCs8xt
Key                     Value
---                     -----
token                   s.YcHxFMgNqhT76U6iERjCs8xt
token_accessor          b0FePlbEGHnEWnfBr3h3qQzN
token_duration          30m
token_renewable         true
token_policies          ["default" "sample-policy"]
identity_policies       []
policies                ["default" "sample-policy"]
```

Module 4: Service Token Lifecycle

4.1 Overview of Lifecycle of Service Tokens

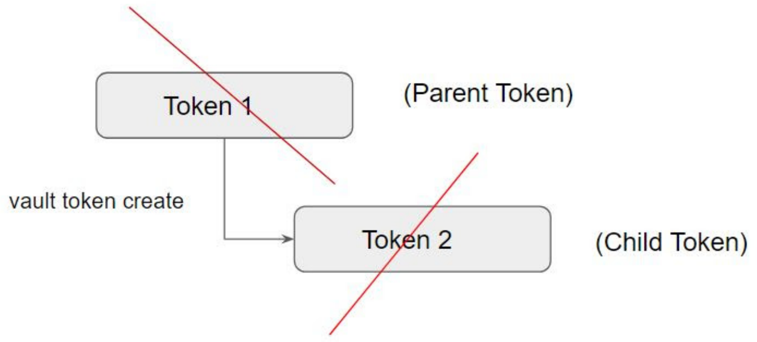
Normally, when a token holder creates new tokens, these tokens will be created as children of the original token



,

4.2 Example Hierarchy

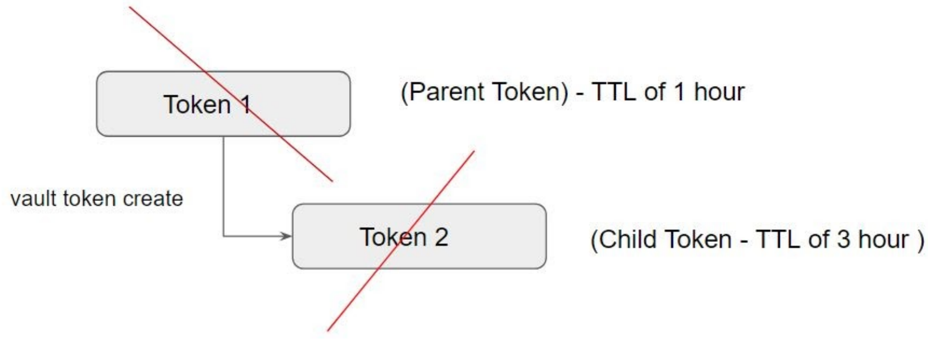
If the parent is revoked or expires, so do all its children regardless of their own TTLs.



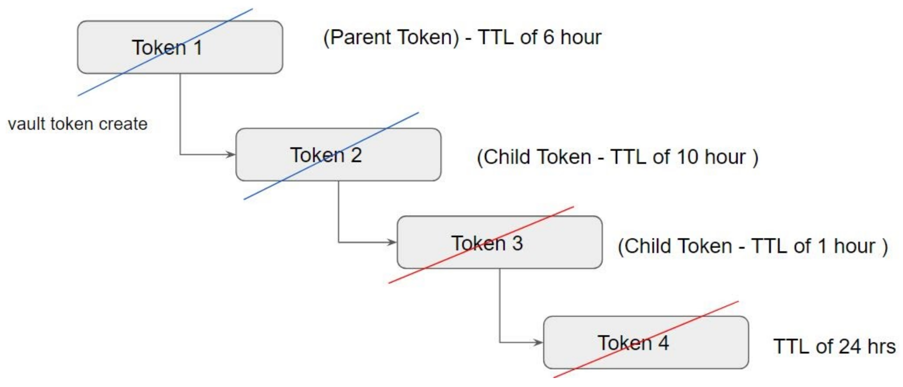
4.3 Example Hierarchy 1 with TTL

In this scenario, Token 1 has a TTL of 1 hour. If it is not renewed, Vault will revoke Token 1.

When Token 1 is revoked, even Token 2 would be removed.


Token 1
(Parent Token) - TTL of 1 hour
vault token create
Token 2
(Child Token - TTL of 3 hour)

4.4 Example Hierarchy 2 with TTL


Token 1
(Parent Token) - TTL of 6 hour
vault token create
Token 2
(Child Token - TTL of 10 hour)
Token 3
(Child Token - TTL of 1 hour)
Token 4
TTL of 24 hrs

Module 5: Token Accessor

When tokens are created, a token accessor is also created and returned.

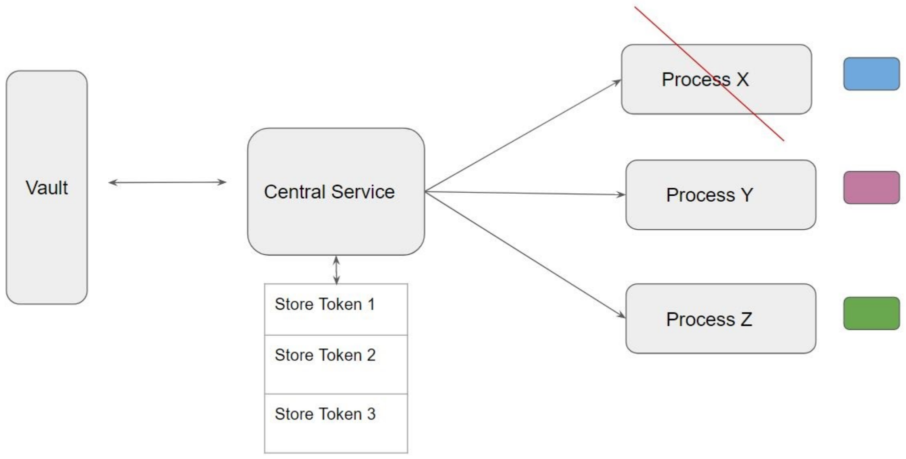
This accessor is a value that acts as a reference to a token and can only be used to perform limited actions:

- Lookup a token's properties (not including the actual token ID)
- Lookup a token's capabilities on a path
- Renew the token
- Revoke the token

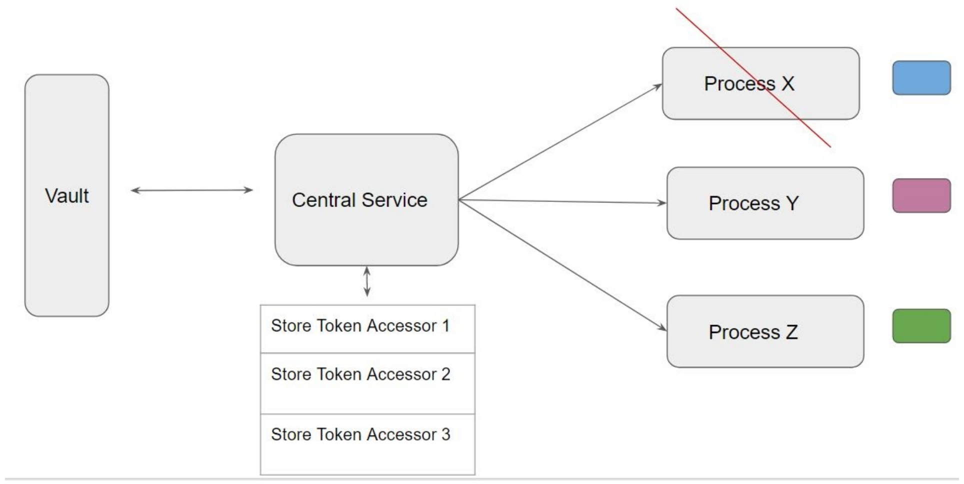
```
C:\Users\Zeal Vora>vault login -method=userpass username=demouser password=password
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                     Value
---                     -----
token                   s.R7q4x4ZGbjSccnSMdPAHuCRC
token_accessor          6UDHnzoYMYgyOkhA26BQYU2J
token_duration          768h
token_renewable         true
token_policies          ["default" "sample-policy"]
identity_policies       []
policies                ["default" "sample-policy"]
token_meta_username     demouser
```

Example Use-Case:



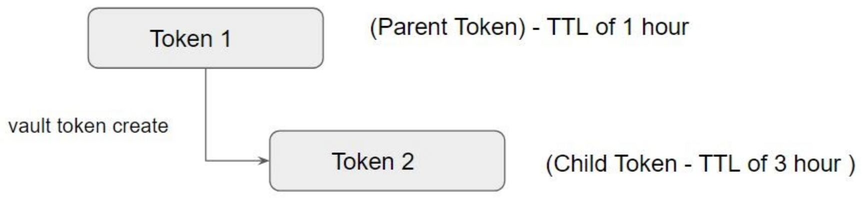
Better Approach:



Module 6: Orphan Tokens

6.1 Overview of Lifecycle of Service Tokens

Normally, when a token holder creates new tokens, these tokens will be created as children of the original token


Token 1
(Parent Token) - TTL of 1 hour
vault token create
Token 2
(Child Token - TTL of 3 hour)

6.2 Overview of Orphan Tokens

Orphan tokens are not children of their parent; therefore, orphan tokens do not expire when their parent does.

They are the root of their own token tree.

Orphan tokens still expire when their own max TTL is reached.

Token 1
(Parent Token)
Orphaned Token 1

Module 7: Secret Engine - Cubby Hole

The cubbyhole secrets engine provides your own private secret storage space where no one else can read (including root).

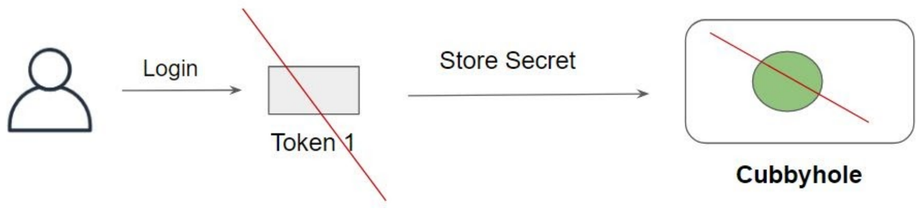
It is not possible to reach into another token's cubbyhole even as the root user.

The cubbyhole secrets engine is enabled by default. It cannot be disabled, moved, or enabled multiple times.

In cubbyhole, paths are scoped per token.

No token can access another token's cubbyhole.

When the token expires, its cubbyhole is destroyed.

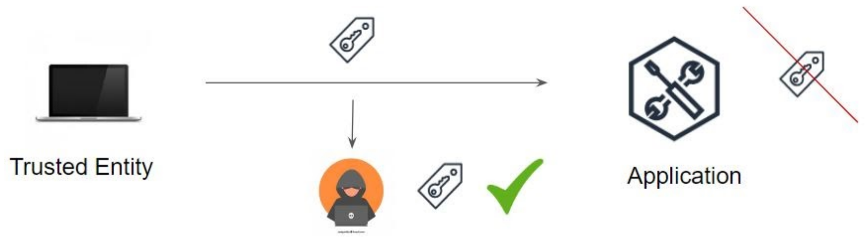


Module 8: Response Wrapping

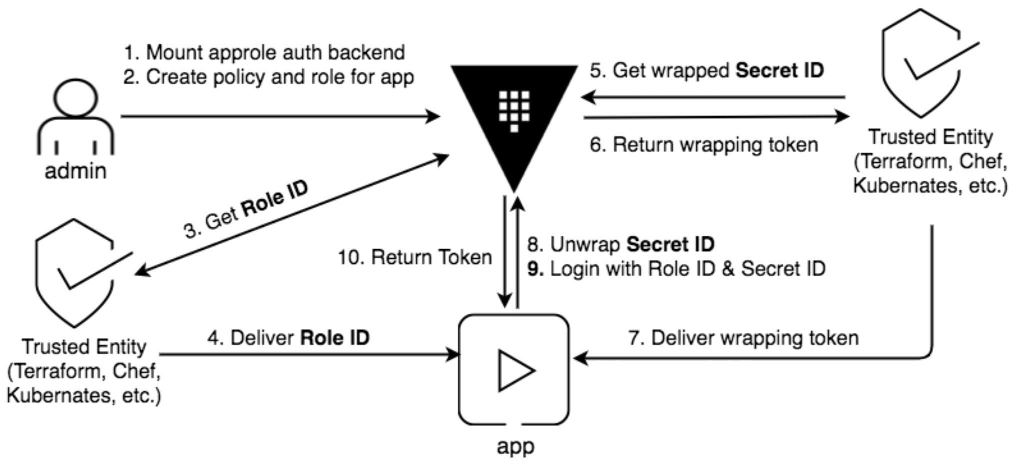
When response wrapping is requested, Vault creates a temporary single-use token (wrapping token) and insert the response into the token's cubbyhole with a short TTL

Only the expecting client who has the wrapping token can unwrap this secret

If the wrapping token is compromised and the attacker unwraps the secret, the application will not be able to unwrap again and this can sound an alarm and you can revoke things accordingly.



The following diagram denotes the workflow of the response wrapping process



1. Mount approle auth backend
2. Create policy and role for app
admin
5. Get wrapped **Secret ID**
6. Return wrapping token
Trusted Entity (Terraform, Chef, Kubernates, etc.)
3. Get **Role ID**
10. Return Token
8. Unwrap **Secret ID**
9. Login with Role ID & Secret ID
Trusted Entity (Terraform, Chef, Kubernates, etc.)
4. Deliver **Role ID**
7. Deliver wrapping token
app

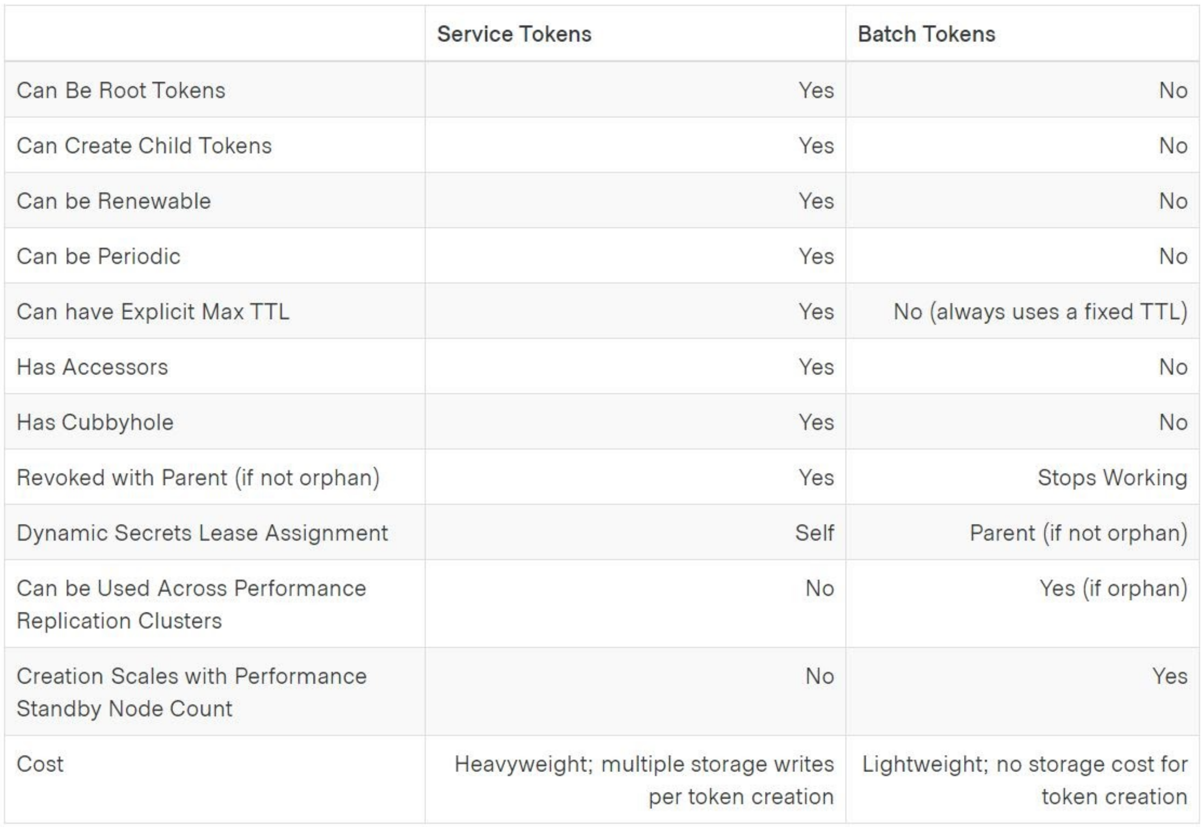
Module 9: Batch Tokens

Batch tokens are encrypted blobs that carry enough information for them to be used for Vault actions, but they require no storage on disk to track them

Even with Vault Replication enabled, the pressure on the storage backend increases as the number of token or lease generation requests increase.

Since Batch Tokens do not require storage on disk and hence are extremely lightweight and scalable but lack most of the flexibility of a service token.

Difference between Service and Batch Tokens:



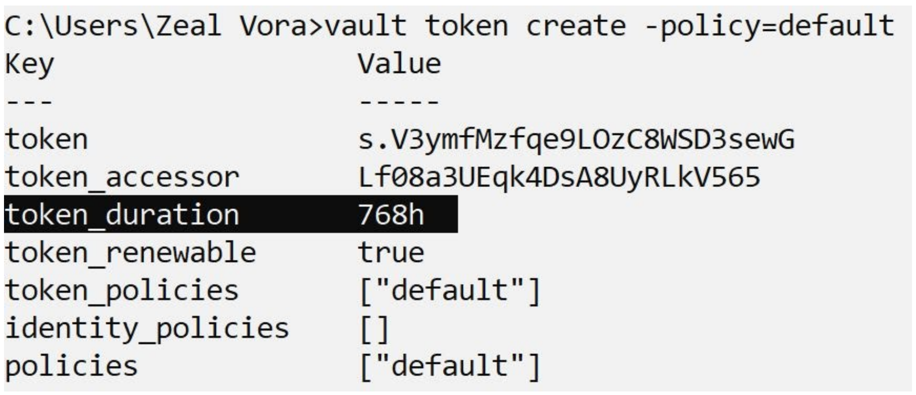
	Service Tokens	Batch Tokens
Can Be Root Tokens	Yes	No
Can Create Child Tokens	Yes	No
Can be Renewable	Yes	No
Can be Periodic	Yes	No
Can have Explicit Max TTL	Yes	No (always uses a fixed TTL)
Has Accessors	Yes	No
Has Cubbyhole	Yes	No
Revoked with Parent (if not orphan)	Yes	Stops Working
Dynamic Secrets Lease Assignment	Self	Parent (if not orphan)
Can be Used Across Performance Replication Clusters	No	Yes (if orphan)
Creation Scales with Performance Standby Node Count	No	Yes
Cost	Heavyweight; multiple storage writes per token creation	Lightweight; no storage cost for token creation

Module 10: Token TTL Configuration

10.1 Revising the Basics

Every non-root token has a time-to-live (TTL) associated with it, which is a current period of validity

After the current TTL is up, the token will no longer function -- it, and its associated leases, are revoked.

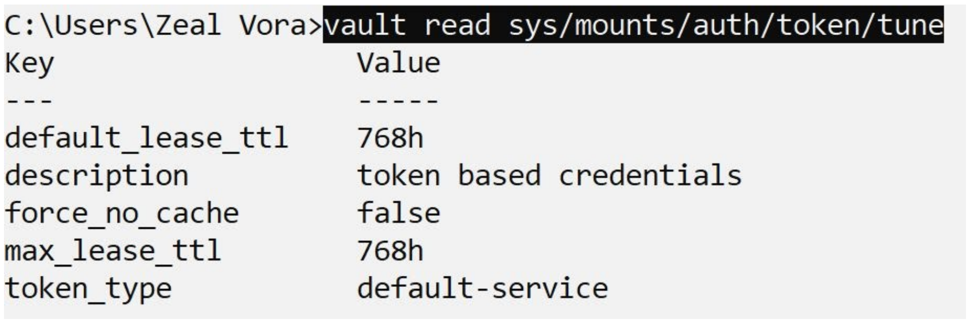


```
C:\Users\Zeal Vora>vault token create -policy=default
Key                  Value
---                  -----
token                s.V3ymfMzfqe9LOzC8WSD3sewG
token_accessor       Lf08a3UEqk4DsA8UyRLkV565
token_duration       768h
token_renewable      true
token_policies       ["default"]
identity_policies    []
policies             ["default"]
```

10.2 TTL Dependency

The TTLs are dependent on a combination of multiple factors. These include:

1. The system max TTL, which is 32 days but can be changed in Vault's configuration file.
2. The max TTL set on a mount using mount tuning
3. A value suggested by the auth method that issued the token.

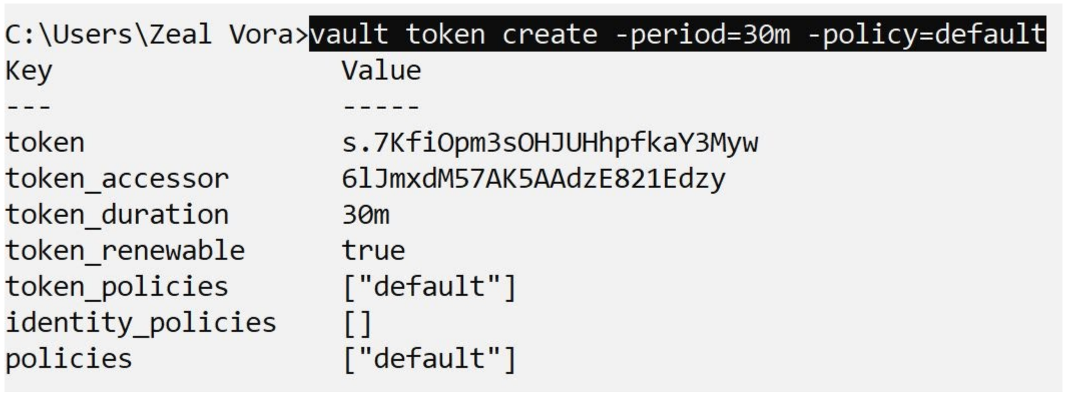


```
C:\Users\Zeal Vora>vault read sys/mounts/auth/token/tune
Key                    Value
---                    -----
default_lease_ttl      768h
description            token based credentials
force_no_cache         false
max_lease_ttl          768h
token_type             default-service
```

Module 11: Periodic Token

Periodic Tokens never expire provided that they are renewed.

Outside of root tokens, it is currently the only way for a token in Vault to have an unlimited lifetime.



```
C:\Users\Zeal Vora>vault token create -period=30m -policy=default
Key                    Value
---                    -----
token                  s.7KfiOpm3sOHJUHhpfkaY3Myw
token_accessor         6lJmxdM57AK5AAdzE821Edzy
token_duration         30m
token_renewable        true
token_policies         ["default"]
identity_policies      []
policies               ["default"]
```